Программный комплекс Систэм Платформ

SePlatform.HMI.Security 2.0

Руководство пользователя

Редакция	Соответствует версии ПО
6. Предварительная	2.0.7

Содержание

1. Руководство пользователя

1.1. О продукте

SePlatform.HMI.Security - это расширение среды разработки и исполнения SePlatform.HMI.

Расширение представляет собой библиотеку компонентов, позволяющих взаимодействовать с подсистемой безопасности SePlatform.Security из проектов SePlatform.HMI.

Под взаимодействием подразумевается:

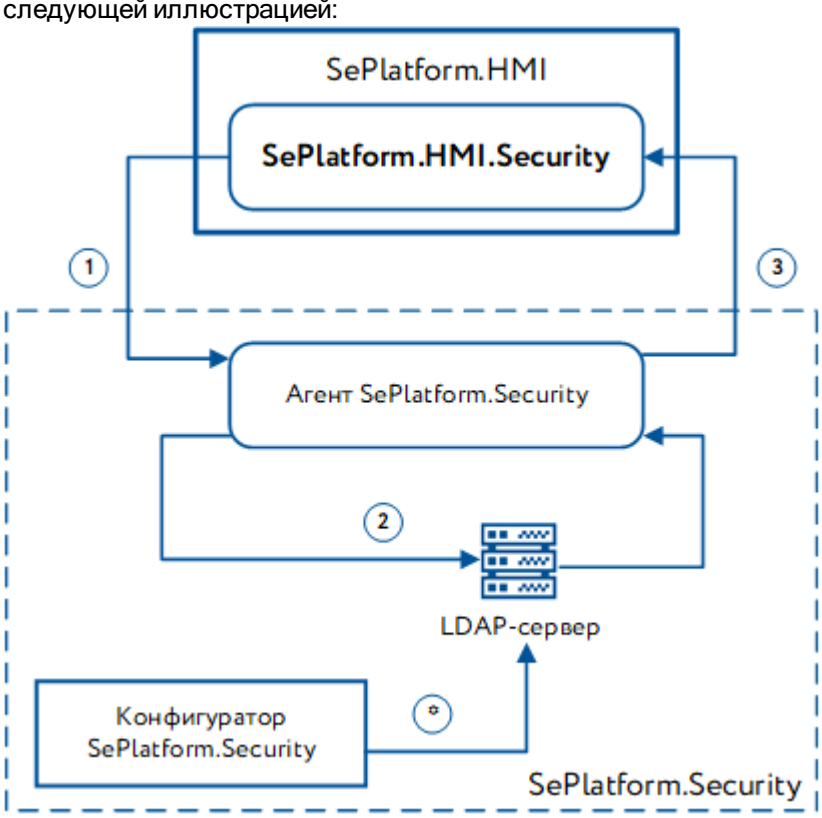
- регистрация пользователя в подсистеме безопасности (вход) с использованием учетных данных;
- просмотр текущей конфигурации подсистемы безопасности SePlatform.Security;
- изменение текущей конфигурации подсистемы безопасности SePlatform.Security;
- получение информации о статусе операций пользователя;
- выход из подсистемы безопасности.

ПРИМЕЧАНИЕ

Компоненты расширения SePlatform.HMI.Security обращаются к подсистеме безопасности SePlatform.Security с помощью API: свойств, функций и событий. Подробнее - в Справочном руководстве (стр. 47).

Место SePlatform.HMI.Security во взаимодействии SePlatform.HMI и SePlatform.Security

Взаимодействие SePlatform.HMI и SePlatform.Security с помощью SePlatform.HMI.Security можно описать следующей иллюстрацией:



1. Компонент SePlatform.HMI.Security отправляет запрос на просмотр или изменение конфигурации подсистемы безопасности в Агент SePlatform.Security;
2. Агент SePlatform.Security находит нужную информацию на LDAP-сервере или записывает новую конфигурацию на LDAP-сервер;
3. Агент SePlatform.Security предоставляет результат операции в проект SePlatform.HMI с помощью компонентов SePlatform.HMI.Security.

ПРИМЕЧАНИЕ

Информация на LDAP-сервере также может быть изменена с помощью Конфигуратор SePlatform.Security.

1.2. Подготовка к работе

1.2.1. Требования к окружению

Для работы расширения SePlatform.HMI.Security должны быть установлены:

- среда разработки и исполнения SePlatform.HMI;
- подсистема безопасности SePlatform.Security;
- компонент SePlatform.Domain - для обеспечения связи между компонентами расширения SePlatform.HMI.Security и подсистемой безопасности SePlatform.Security.

Если работаете с веб-версией проекта автоматизации, установите дополнительно SePlatform.HMI.WebViewer - для использования проектов SePlatform.HMI в веб-интерфейсе.

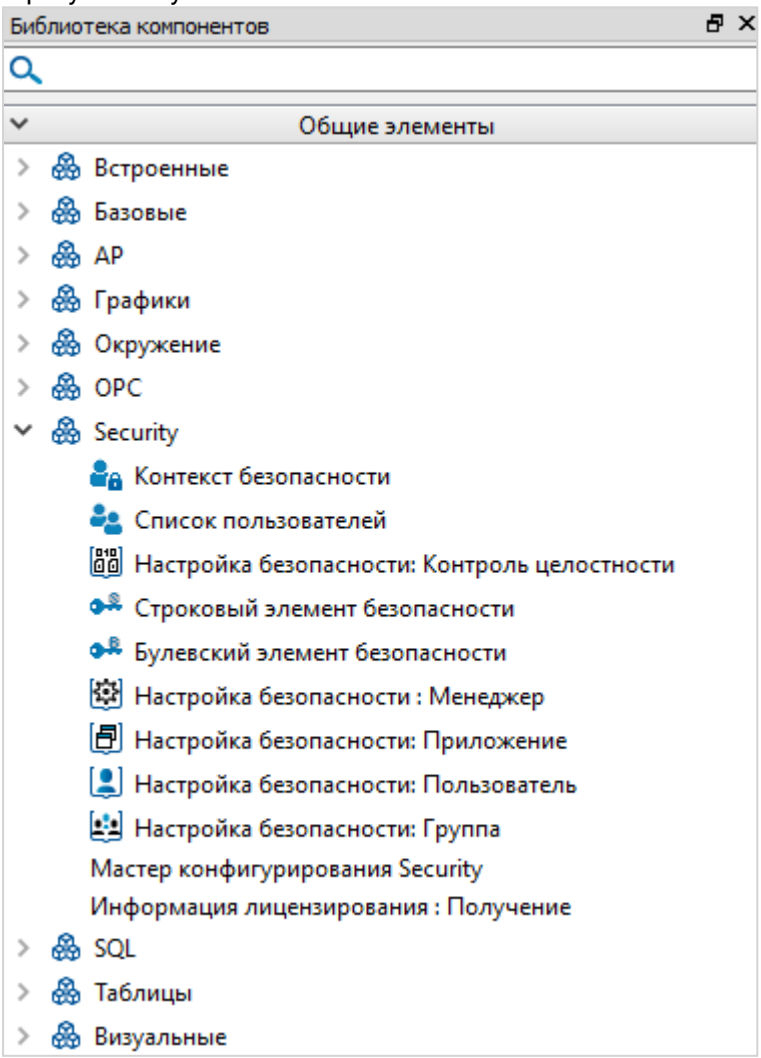
1.2.2. Установка, удаление или восстановление

ОС Windows

Для установки, удаления или восстановления SePlatform.HMI.Security запустите установочный файл `seplatform.hmi.security-<lng>-<arch>-<version>.msi` и следуйте инструкциям мастера. Если устанавливаете веб-версию, запустите файл `seplatform.hmi.security.webviewer-<lng>-<arch>-<version>.msi`.

> ПРИМЕЧАНИЕ
>
> В названии файла `<lng>` - это язык расширения, `<arch>` - целевая процессорная архитектура, а `<version>` - номер версии расширения.

В результате установки в библиотеке компонентов SePlatform.HMI должен появиться юнит **`Security`**.



ОС Linux

ОБРАТИТЕ ВНИМАНИЕ

Команды на установку и удаление нужно запускать с правами суперпользователя.

Для установки SePlatform.HMI.Security вызовите пакет с командой на установку в зависимости от используемого пакетного менеджера:

- rpm-пакет:

```
sudo rpm -i seplatform.hmi.security-<lng>-<version>.rpm
```

или, если используете веб-версию:

```
sudo rpm -i seplatform.hmi.security.webviewer-<lng>-<version>.rpm
```

- deb-пакет:

```
sudo dpkg -i seplatform.hmi.security-<lng>-<version>.deb
```

или, если используете веб-версию:

```
sudo dpkg -i seplatform.hmi.security.webviewer-<lng>-<version>.deb
```

ПРИМЕЧАНИЕ

В названии пакета `<lng>` - это язык компонента, а `<version>` - номер версии расширения.

В результате установки в библиотеке компонентов SePlatform.HMI должен появиться юнит **`Security`**.
Для удаления SePlatform.HMI.Security вызовите команду на удаление приложения в зависимости от используемого пакетного менеджера:

- rpm-пакет:

```
sudo rpm -e seplatform.hmi.security-desktop
```

или, если используете веб-версию:

```
sudo rpm -e seplatform.hmi.security-webviewer
```

- deb-пакет:

```
sudo dpkg -r seplatform.hmi.security-desktop
```

или, если используете веб-версию:

```
sudo dpkg -r seplatform.hmi.security-webviewer
```

1.3. Управление доступом в проектах

ОБРАТИТЕ ВНИМАНИЕ

Прежде чем изучить этот и следующие разделы рекомендуется ознакомиться с:

- основами разработки проектов в Дизайнер SePlatform.HMI;
- назначением и принципом работы подсистемы безопасности SePlatform.Security;
- принципами конфигурирования подсистемы безопасности SePlatform.Security.

В разделе продемонстрировано применение компонентов SePlatform.HMI.Security для разграничения доступа пользователей в проектах SePlatform.HMI. Для этого описан проект-пример, имитирующий АРМ системы управления насосом и резервуаром. Доступ к управлению имитируемой системой регулируется подсистемой безопасности SePlatform.Security и зависит от прав текущего пользователя.

Для работы с проектом-примером должны быть установлены:

- Дизайнер SePlatform.HMI - для просмотра, запуска и модификации проекта;
- SePlatform.Security - для создания учетных записей пользователей и прав;
- SePlatform.Domain - для обеспечения взаимодействия компонентов подсистемы безопасности SePlatform.Security;
- SePlatform.HMI.Security - для взаимодействия SePlatform.HMI и SePlatform.Security.

Также для работы необходимы:

- файлы проекта-примера `AccessControl.hmi`;
- файл конфигурации SePlatform.Security `Configuration.Example.bak`.

Скачайте архив с указанными файлами. Откройте проект-пример в Дизайнер SePlatform.HMI. Восстановите резервную копию из файла конфигурации SePlatform.Security.

ВАЖНО

Если у вас имеется собственная конфигурация SePlatform.Security, сохраните ее резервную копию, прежде чем приступить к работе с проектом-примером.

Убедитесь, что в SePlatform.Security появились:

- учетные записи пользователей **«Иванов»** и **«Петров»**, состоящих в группах **«Диспетчеры»** и **«Операторы»** соответственно;
- приложение **«Управление состоянием оборудования»**, содержащее:
  - два логических права: **«Управление насосом»** и **«Управление резервуаром»**;
  - одно строковое право **«Доступ к оборудованию»**.

Назначение проекта-примера

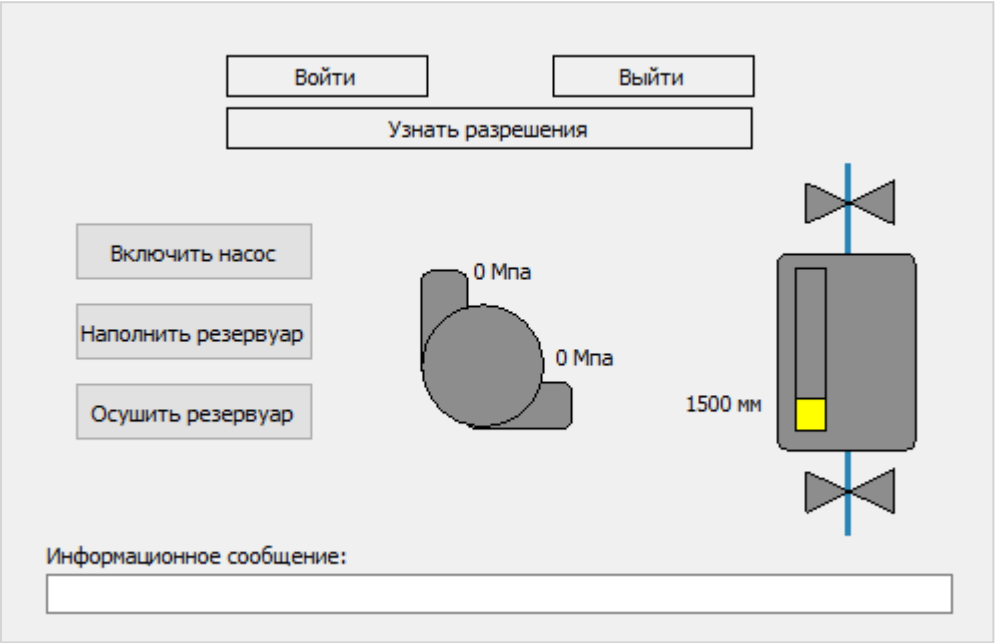
Проект-пример имитирует АРМ, которое позволяет:

- запустить проект под пользователем по умолчанию и воспользоваться общедоступными функциями АРМ без ввода учетных данных;
- войти с личными учетными данными (логин/пароль);
- воспользоваться функциями, доступными только одному из пользователей:
  - пользователю, состоящему в группе **«Диспетчеры»**, доступно управление резервуаром, но недоступно управление состоянием насоса;

- пользователю, состоящему в группе **«Операторы»**, доступно управление состоянием насоса, но недоступно управление резервуаром;
- выйти, не потеряв возможности воспользоваться общедоступными функциями.

Состав проекта-примера

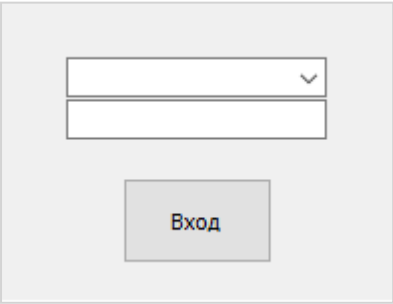
Главная экранная форма проекта - *MainForm*:



На главную форму добавлены:

- *SecurityContext* - элемент типа **Контекст безопасности** (стр. 47), обеспечивающий взаимодействие с SePlatform.Security;
- *BoolTokenProxy_ControlTank* - элемент типа **Булевский элемент безопасности** (стр. 74), привязанный к праву, регулирующему доступ к управлению резервуаром;
- *BoolTokenProxy_ControlPump* - элемент типа **Булевский элемент безопасности**, привязанный к праву, регулирующему доступ к управлению насосом;
- *StringTokenProxy* - элемент типа **Строковый элемент безопасности** (стр. 72), привязанный к праву, содержащему текстовые разрешения и запреты обоих пользователей.

Вторая форма проекта - форма входа *AuthorizationForm*:



На форму входа *AuthorizationForm* добавлены:

- *UserList* - элемент типа **Список пользователей** (стр. 60), позволяющий работать со списком учетных записей из подсистемы безопасности SePlatform.Security;
- *SecurityContext* - элемент типа **Контекст безопасности**, обеспечивающий взаимодействие с SePlatform.Security.

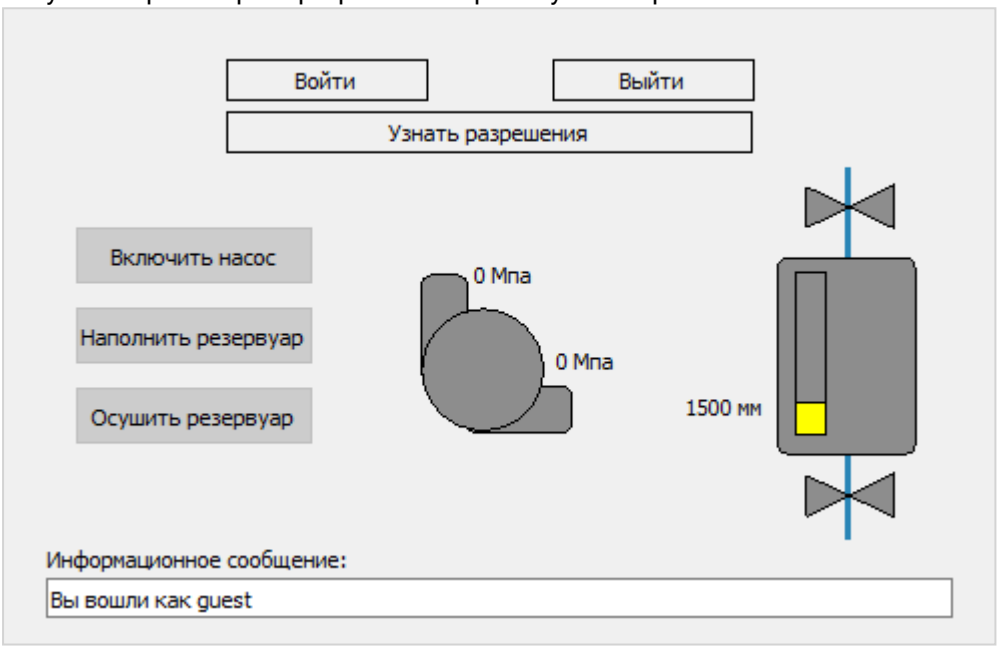
ПРИМЕЧАНИЕ

Здесь и в других проектах-примерах на каждой форме размещен свой элемент типа **Контекст безопасности**. Такое решение не является обязательным. Можно применить любое подходящее для проекта решение, например:

- использовать ссылки на элемент типа **Контекст безопасности**, расположенный на одной из форм проекта;
- использовать один глобальный элемент типа **Контекст безопасности**.

Использование проекта с правами пользователя по умолчанию

Запустите проект-пример в рантайм. При запуске откроется главное окно:



При запуске главного окна элемент *SecurityContext* обратился к Агент SePlatform.Security для получения информации о текущем пользователе и его правах. В систему никто не вошел, поэтому Агент SePlatform.Security предоставил информацию о пользователе, чья учетная запись используется по умолчанию - **«guest»**.

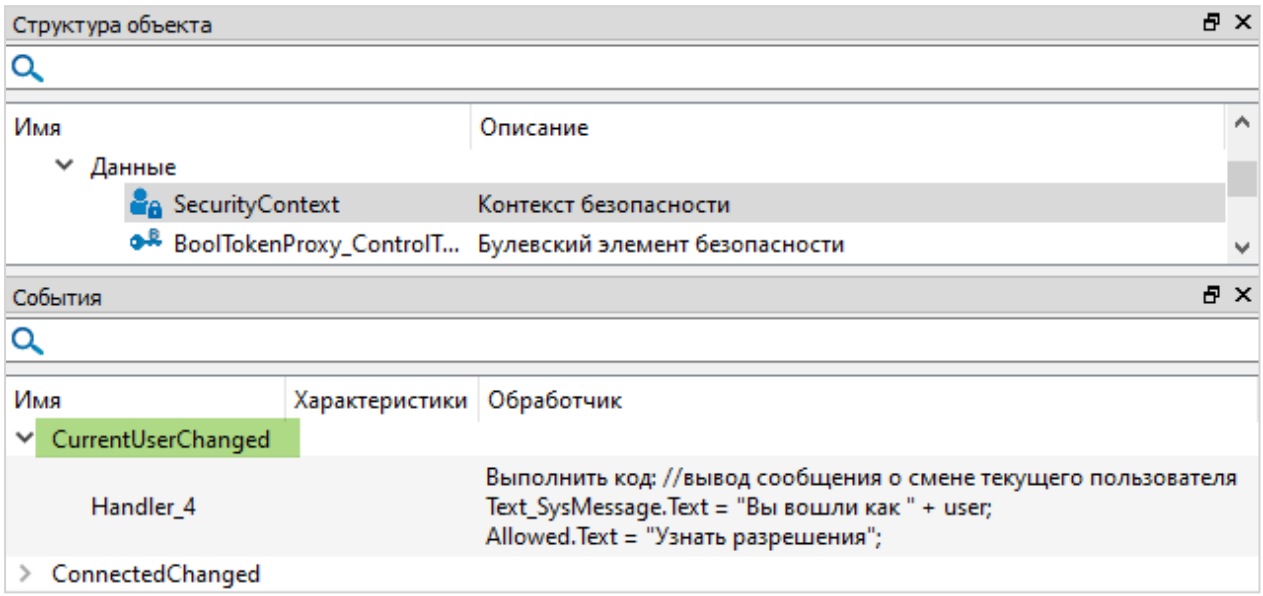
ПРИМЕЧАНИЕ

В качестве пользователя по умолчанию можно указать любого из имеющихся в подсистеме безопасности SePlatform.Security.

Пользователь **«guest»** не имеет прав на управление системой, поэтому кнопки **Включить насос**, **Наполнить резервуар** и **Осушить резервуар** недоступны. О том, что текущим пользователем является **«guest»**, написано в информационном сообщении в нижней части главного окна.

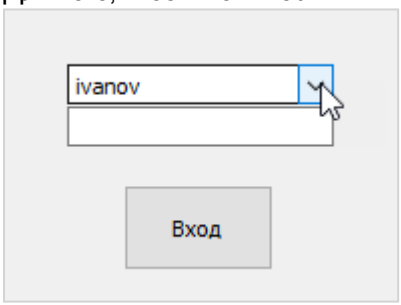
ПРИМЕЧАНИЕ

Тексты сообщений о состоянии подключения помещены в обработчики событий элемента *SecurityContext*. Ниже приведен код обработчика события **CurrentUserChanged** (стр. 58), активирующегося при смене текущего пользователя:



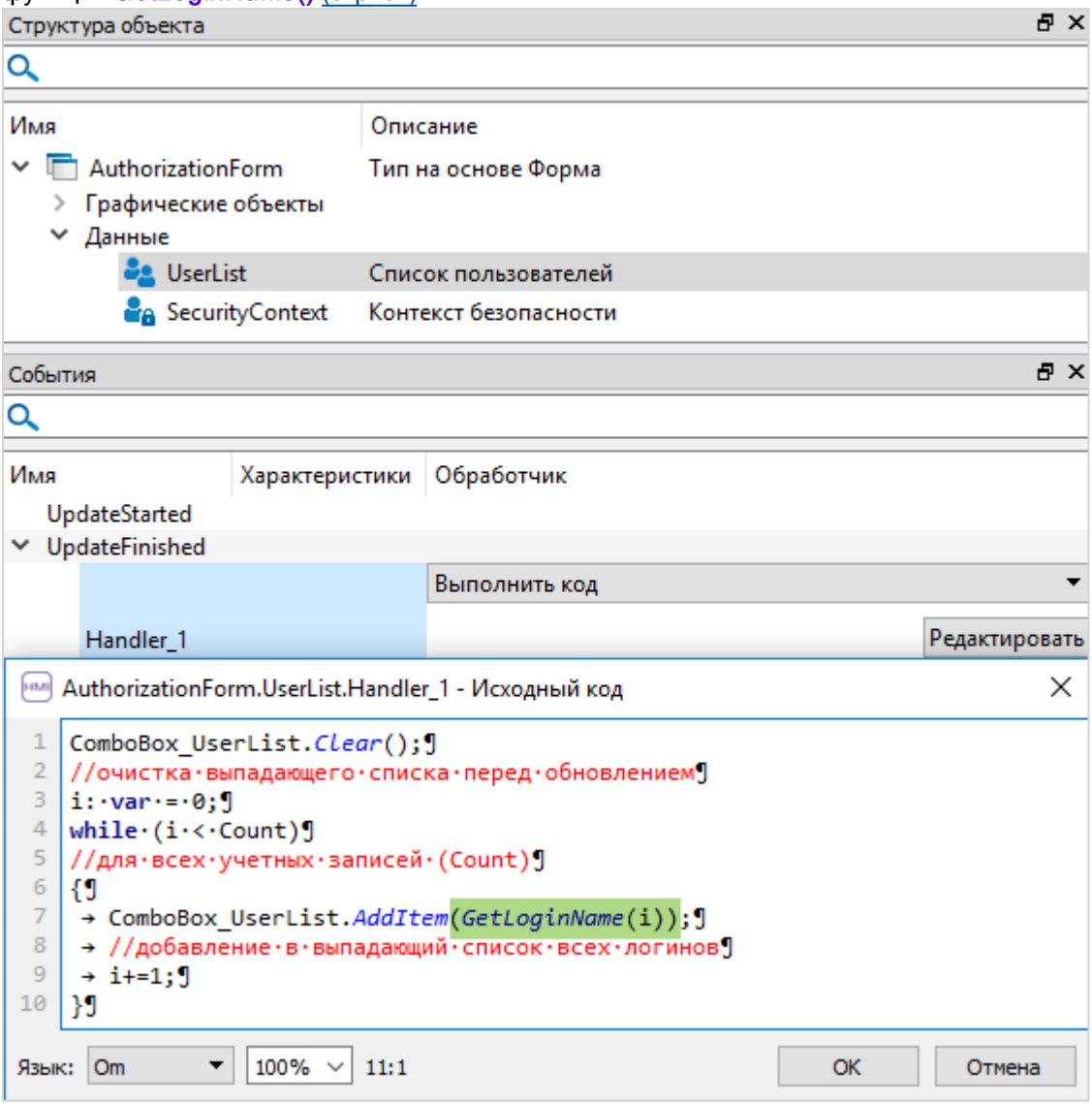
Вход с личными учетными данными

Для того, чтобы войти с личными учетными данными, нажмите кнопку **Войти**. Откроется окно входа:



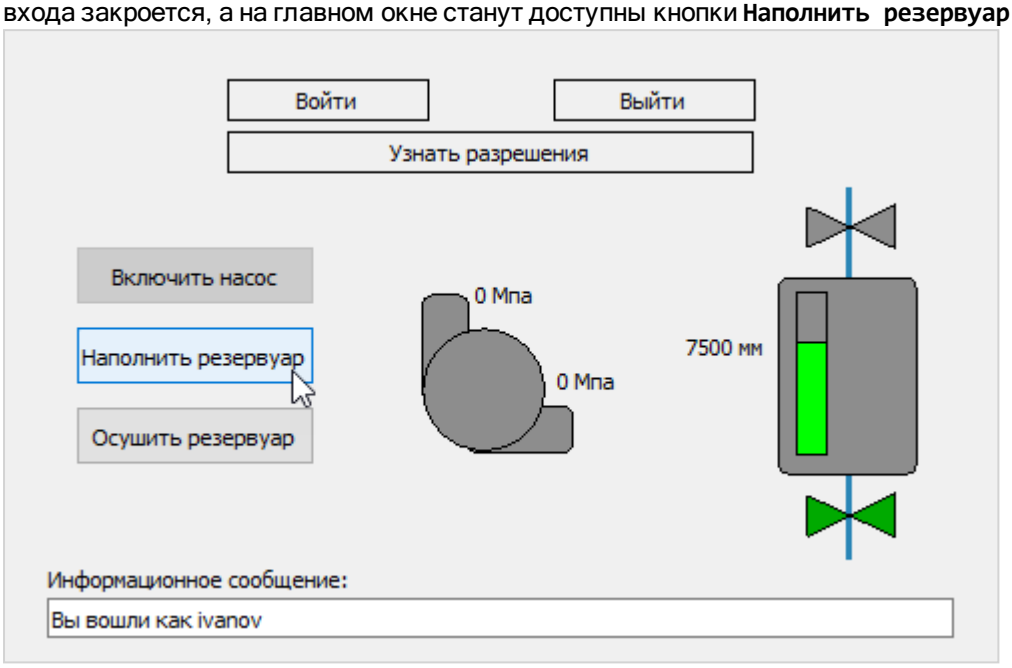
Логин можно ввести либо выбрать из выпадающего списка.

За предоставление списка учетных записей отвечает элемент *UserList*. При запуске окна входа с помощью функции **BeginUpdate()** (стр. 63) был запрошен список учетных записей из подсистемы безопасности SePlatform.Security (см. код обработчика события **Opened** формы входа). Когда обновление завершилось, активировалось событие **UpdateFinished** (стр. 63) элемента *UserList*. Ознакомьтесь с кодом обработчика этого события, написанном на языке SePlatform.Om. Для получения логинов учетных записей используется функция **GetLoginName()** (стр. 62):



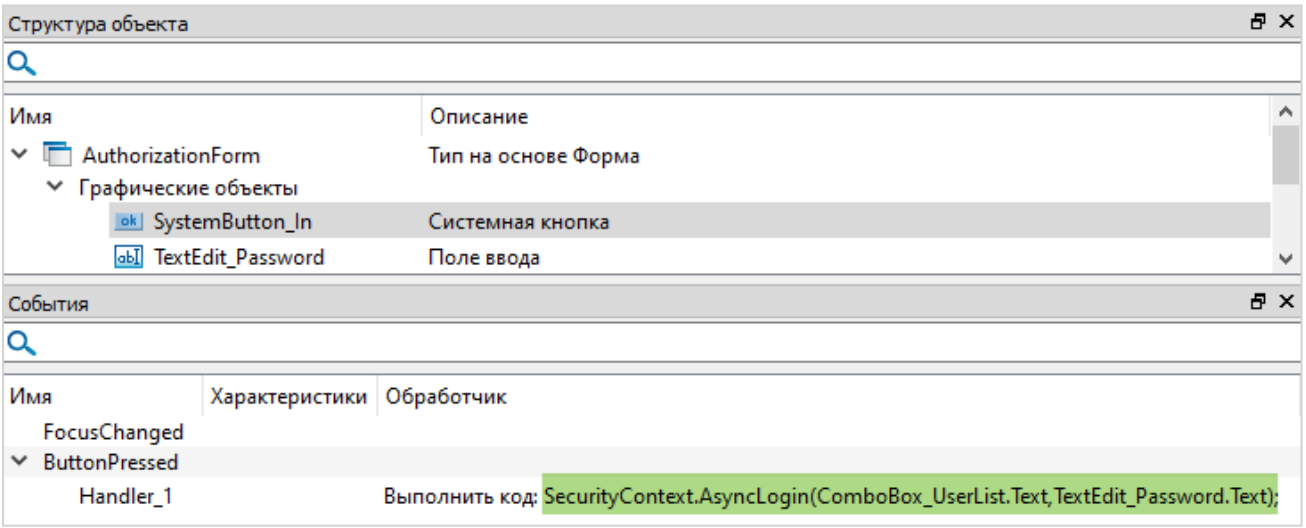
Использование проекта с личной учетной записью

Выберите логин **«ivanov»** из выпадающего списка и введите пароль **«ivanov1»**. Нажмите кнопку **Вход**. Окно входа закроется, а на главном окне станут доступны кнопки **Наполнить резервуар** и **Осушить резервуар**:



При нажатии кнопки **Вход**:

1. Была вызвана функция AsyncLogin() (стр. 53) элемента *SecurityContext*, входными параметрами для которой являются логин и пароль:

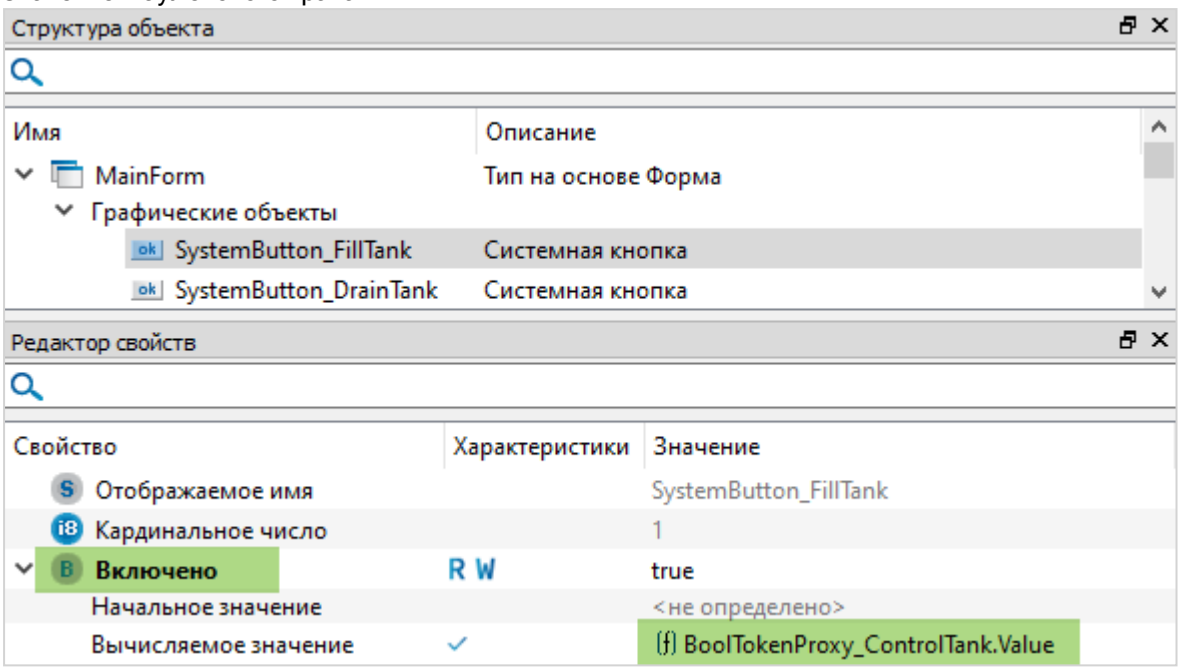


2. Агент SePlatform.Security принял введенные учетные данные и зарегистрировал **«ivanov»** как текущего пользователя.

3. Агент SePlatform.Security обновил текущие значения прав в соответствии с конфигурацией SePlatform.Security для текущего пользователя.

4. Элементы *BoolTokenProxy_ControlTank*, *BoolTokenProxy_ControlPump* и *StringTokenProxy* получили информацию о новых значениях прав.

5. Кнопки управления резервуаром стали доступны, так как их свойства активности связаны со значением булевского права:



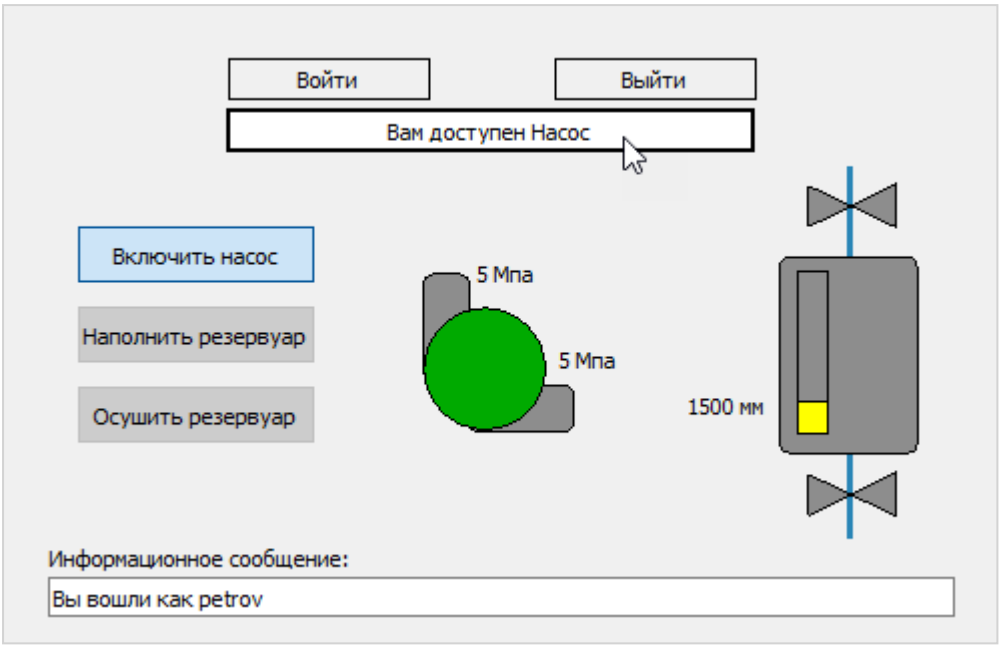
Нажмите кнопку **Узнать разрешения**. Название кнопки изменится. Теперь на ней будет написано, какое оборудование доступно пользователю. Ознакомьтесь с обработчиком события MouseClick этой кнопки. Элемент *StringTokenProxy* предоставляет значение разрешения текущего пользователя с помощью функции **GetAllowed()** (стр. 73).

Нажмите кнопку **Выйти**. Будет вызвана функция **LogOut()** (стр. 54) элемента *SecurityContext*. В информационном сообщении вы увидите, что текущим пользователем снова стал **«guest»**, а кнопки управления резервуаром недоступны.

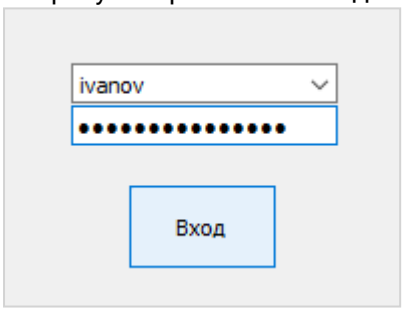
Откройте окно входа и введите другие учетные данные: логин **«petrov»** и пароль **«petrov1»**.

1. Вновь будет вызвана функция **AsyncLogin()** элемента *SecurityContext*, и на стороне Агент SePlatform.Security **«petrov»** будет зарегистрирован как текущий пользователь.
2. Выполнится условие активности, связанное с правом на управление насосом, и кнопка **Включить насос** станет доступна.

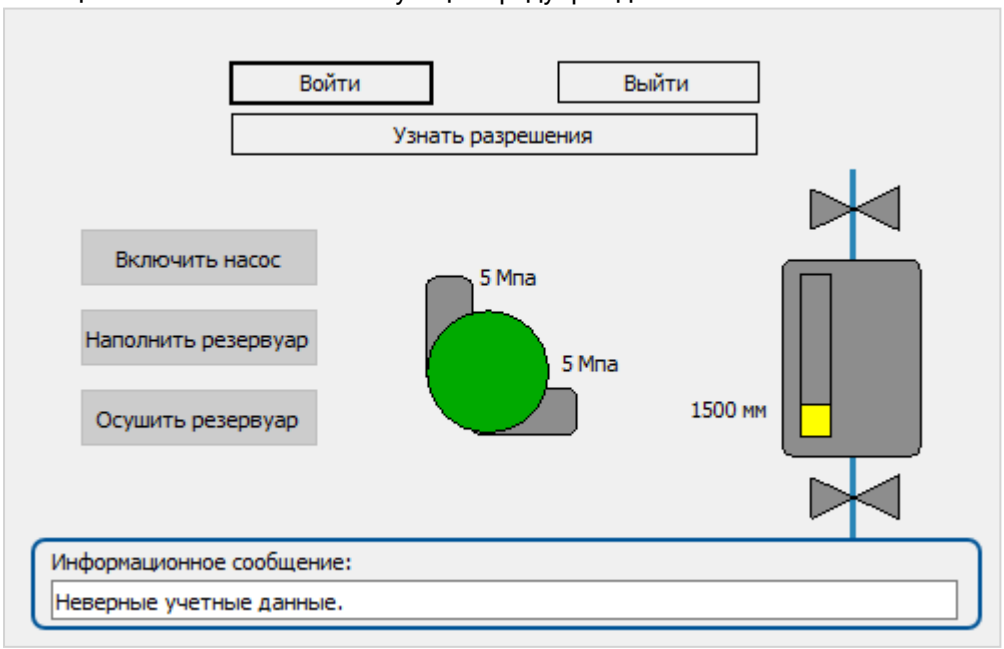
3. При нажатии на кнопку **Узнать разрешения** появится текстовое описание разрешения текущего пользователя:



Попробуйте при попытке входа ввести случайную комбинацию символов в качестве логина и/или пароля:



Так как введены несуществующие учетные данные, Агент SePlatform.Security отклонит попытку входа. Активируется событие **LoginRejected** (стр. 59) элемента *SecurityContext*, и в поле вывода информационного сообщения появится соответствующее предупреждение:



1.4. Собственная форма входа

Когда пользователь вводит личные учетные данные в проекте SePlatform.HMI, подсистема безопасности фиксирует его как текущего пользователя, и может предоставлять информацию о нем сторонним приложениям. Для удобства регистрации в подсистеме безопасности можно использовать собственную форму входа.

Для создания собственной формы входа необходимы компоненты расширения SePlatform.HMI.Security. В разделе описано, как с их помощью:

- использовать ограничение длительности сессий пользователей;
- использовать ограничение срока действия паролей;
- менять пароли в учетных записях.

ПРИМЕЧАНИЕ

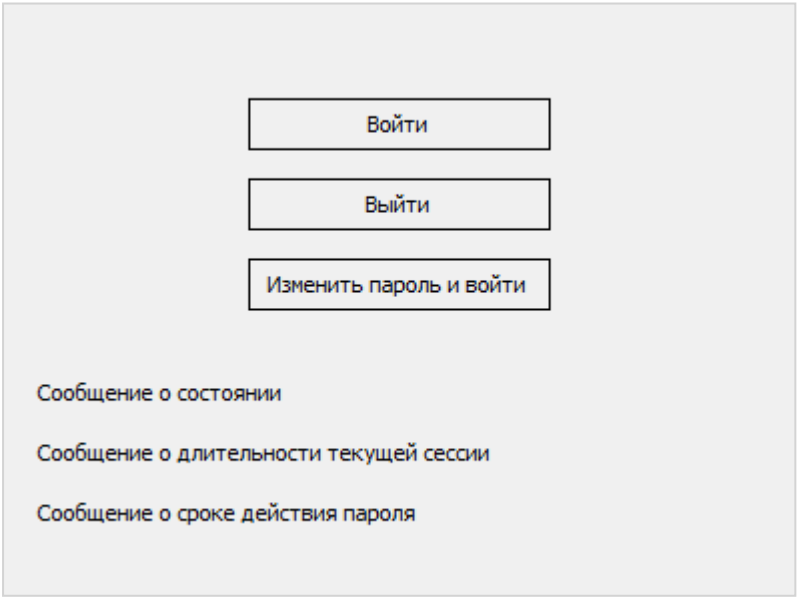
Раздел описывает расширенный сценарий применения компонентов SePlatform.HMI.Security. В большинстве случаев в проектах SePlatform.HMI достаточно обеспечить вход и выход пользователя так, как описано в разделе 1.3. Управление доступом в проектах (стр. 9).

Для наглядности описан проект-пример формы входа.

1. Скачайте архив с проектом-примером `AuthorizationForm.hmi` и откройте в Дизайнер SePlatform.HMI.
2. Запустите Конфигуратор SePlatform.Security для просмотра текущей конфигурации подсистемы безопасности.

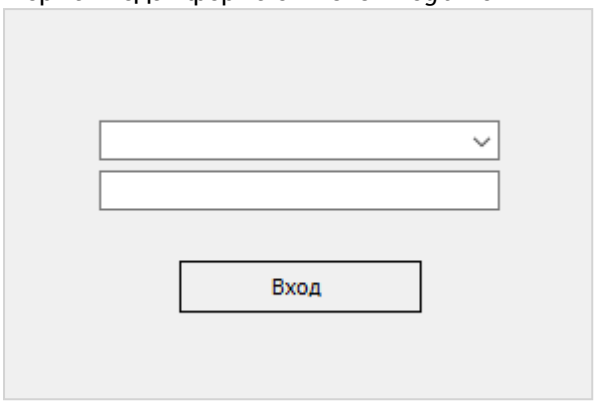
Состав проекта-примера

Основная форма проекта-примера - *`AuthorizationForm`*:



Форма *`AuthorizationForm`* предназначена для вывода информационных сообщений. На ней расположены элементы управления, поля вывода системных сообщений и два элемента типа **Таймер**. Кроме того, на форме находится *`SecurityContext`* - элемент типа **Контекст безопасности** (стр. 47), обеспечивающий взаимодействие с SePlatform.Security.

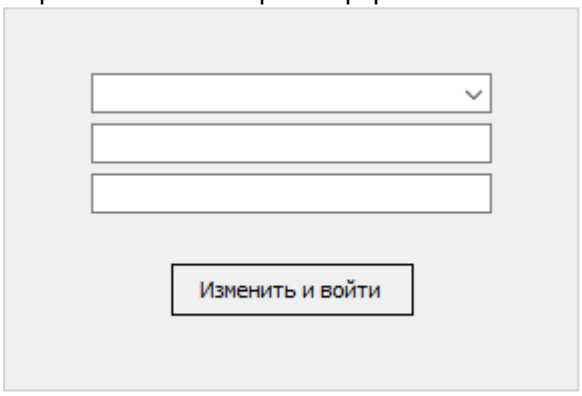
Форма входа - форма с именем *LoginForm*:



Форма *LoginForm* предназначена для ввода учетных данных. На форму добавлены:

- *SecurityContext* - элемент типа **Контекст безопасности**, обеспечивающий взаимодействие с подсистемой безопасности;
- *UserList* - элемент типа **Список пользователей** (стр. 60), позволяющий работать со списком учетных записей из подсистемы безопасности.

Форма изменения пароля - форма с именем *ChangePasswordAndLoginForm*:



Форма предназначена для изменения пароля и регистрации в подсистеме безопасности с новыми учетными данными. На форму добавлены:

- *SecurityContext* - элемент типа **Контекст безопасности**, обеспечивающий взаимодействие с подсистемой безопасности;
- *UserList* - элемент типа **Список пользователей**, позволяющий работать со списком учетных записей из подсистемы безопасности.

Уведомление об ограничении длительности сессии

Для того, чтобы ограничить длительность сессии пользователя, необходимо настроить соответствующее право **Максимальное время сессии** (**SessionDurationLimit**) в Конфигуратор SePlatform.Security. Обратите внимание на вкладку учетной записи пользователя **«Иванов»**: для него это право уже добавлено, длительность сессии - 2 минуты. По истечении двух минут после входа сессия завершается, и текущим пользователем становится пользователь по умолчанию.

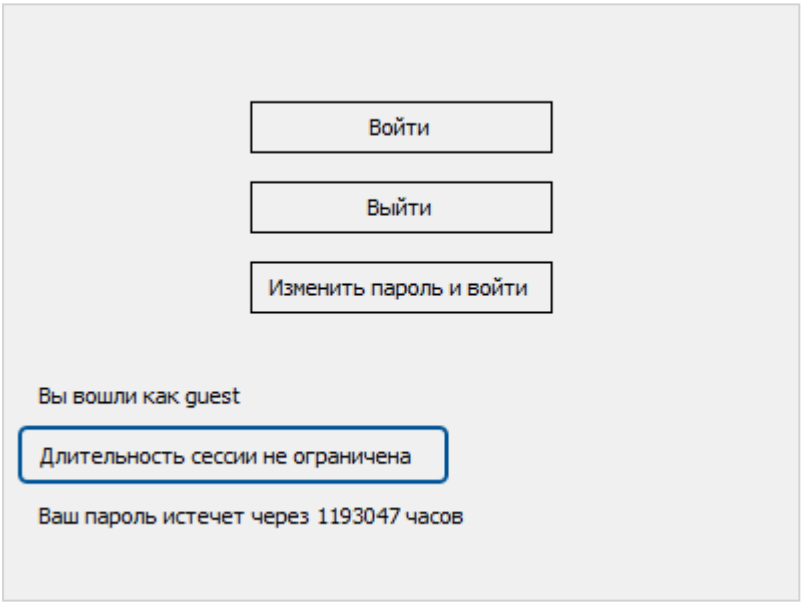
С правом **Максимальное время сессии** (**SessionDurationLimit**) связаны свойства элемента *SecurityContext*:

- SessionDurationLimit (стр. 49) - хранит значение, записанное в право **Максимальное время сессии** (**SessionDurationLimit**);
- SessionExpiresIn (стр. 50) - хранит значение времени, оставшегося до завершения сессии текущего пользователя.

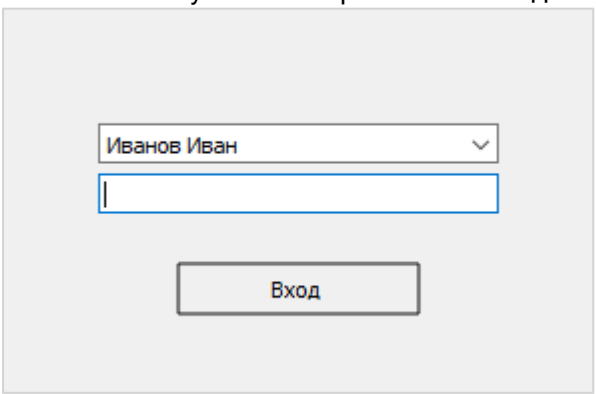
Откройте форму входа *LoginForm*. Ознакомьтесь с кодом обработчика события OnTimer элемента *Timer_Session*, написанном на языке SePlatform.Om. Свойство SessionExpiresIn применяется для вывода сообщения об оставшемся до завершения сессии времени:

```
if (SecurityContext.SessionExpiresIn == 0)
//если время до истечения сессии пользователя равно 0, сессия не ограничена
{
 Text_SysMsgSessionDuration.Text = "Длительность сессии не ограничена";
}
else
{
 //в противном случае выводится значение времени, оставшегося до истечении сессии,
переведенное в минуты
 Text_SysMsgSessionDuration.Text="До конца сессии осталось "
 +String.ToString(SecurityContext.SessionExpiresIn/60+1)+" минут(-ы)";
}
```

Запустите проект в рантайм. Откроется окно выбора действия - основное окно проекта. Для пользователя по умолчанию **«guest»** длительность сессии не ограничена, о чем будет написано в информационном сообщении:

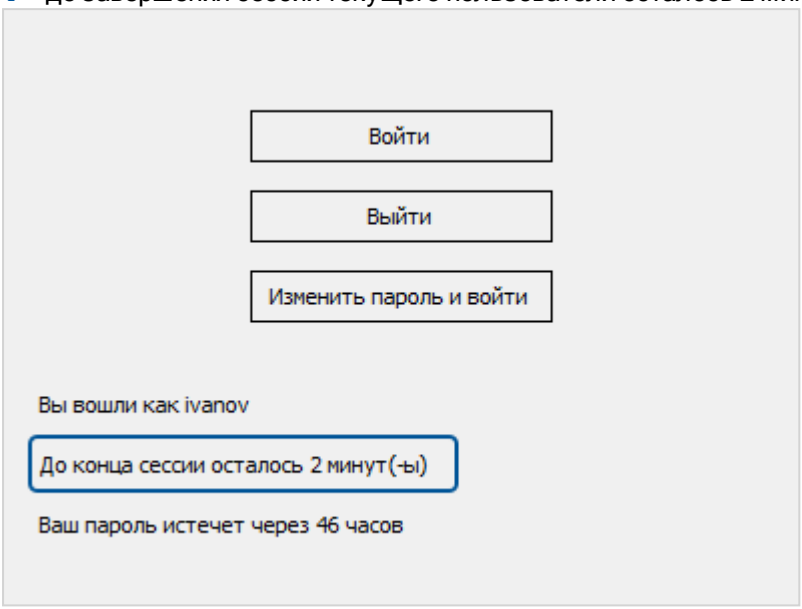


Нажмите кнопку **Войти**. Откроется окно входа:



Из выпадающего списка выберите имя **«Иванов»** и введите пароль **«ivanov1»**. Окно входа закроется, а в информационном сообщении основного окна появится уведомление о том, что:

- текущим пользователем стал пользователь с логином **«ivanov»**;
- до завершения сессии текущего пользователя осталось 2 минуты:



ПРИМЕЧАНИЕ

О том, как добавить в выпадающий список логины учетных записей, подробнее написано в разделе 1.3. Управление доступом в проектах (стр. 9).

При вводе учетных данных видеть имя в выпадающем списке привычнее, чем логин. Поэтому в данном примере выпадающий список заполняется не логинами, а отображаемыми именами. Для этого вместо функции **GetLoginName()** (стр. 62) используется функция **GetDisplayName()** (стр. 62) элемента *UserList*. Функция **GetDisplayName()** возвращает отображаемое имя учетной записи.

Уведомление об истечении срока действия пароля

Для того, чтобы установить срок действия пароля учетной записи, необходимо настроить право **Срок действия пароля** (**PasswordAge**) в Конфигуратор SePlatform.Security. Для пользователя с именем **«Иванов»** максимальный срок действия пароля - 2 дня. Через два дня после изменения пароля попытки входа отклоняются подсистемой безопасности.

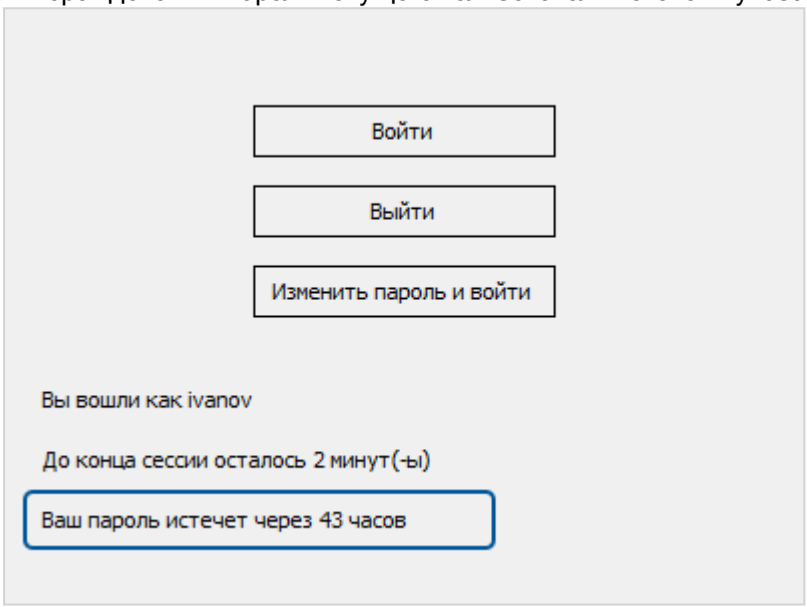
С правом **Срок действия пароля** (**PasswordAge**) связано свойство **PasswordExpiresIn** (стр. 51) элемента *SecurityContext*. Это свойство хранит оставшееся время действия пароля.
Откройте форму входа *LoginForm*. Ознакомьтесь с кодом обработчика события **OnTimer** элемента *Timer_Password*, написанном на языке SePlatform.Om. Свойство **PasswordExpiresIn** применяется для вывода сообщения об истечении срока действия пароля:

```
if (SecurityContext.PasswordExpiresIn == 0)
//если время до истечения срока действия пароля равно 0, срок действия пароля не ограничен
{
 Text_SysMsgPswdExpiration.Text = "Срок действия пароля не ограничен";
}
else
{
```

```
 //в противном случае выводится значение времени, оставшегося до истечении срока действия
пароля, переведенное в часы
 Text_SysMsgPswdExpiration="Ваш пароль истечет через "
 +String.ToString(SecurityContext.PasswordExpiresIn/3600+1)+" часов";
}
```

Запустите проект в рантайм. Войдите с учетными данными **«Иванов»**. В информационном сообщении основного окна появится уведомление о том, что:

- текущим пользователем стал пользователь с логином **«ivanov»**;
- срок действия пароля текущего пользователя истечет в указанное время:



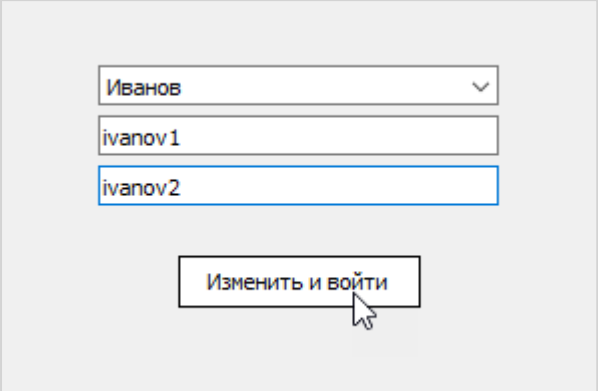
Изменение пароля учетной записи

Когда необходимо реализовать возможность изменения пароля, можно использовать функцию AsyncLoginWithPasswordChange() (стр. 53) элемента *SecurityContext*.
Откройте форму изменения пароля *ChangePasswordAndLoginForm*. Ознакомьтесь с обработчиком события MouseClick кнопки **Изменить и войти**:

```
SecurityContext.AsyncLoginWithPasswordChange(UserName[ComboBox_
UserList.SelectedIndex],TextEdit_OldPswd,TextEdit_NewPswd);
//выполнение функции AsyncLoginWithPasswordChange, входными параметрами для которой являются
логин, старый пароль и новый пароль
Close();
```

Входными параметрами функции AsyncLoginWithPasswordChange() являются логин, текущий и новый пароль, введенные в соответствующие поля.

Запустите проект в рантайм и нажмите кнопку **Изменить пароль и войти**. Откроется окно изменения пароля. Выберите из выпадающего списка имя учетной записи **«Иванов»**, введите текущий пароль учетной записи **«ivanov1»**, а затем придумайте и введите новый пароль. Нажмите кнопку **Изменить и войти**:



Окно изменения пароля закроется, а результат операции появится в информационном сообщении основного окна. Если операция завершится без ошибок, будет осуществлен вход с новыми учетными данными.

ОБРАТИТЕ ВНИМАНИЕ

Пароль не удастся изменить, если не вышел минимальный срок его действия. Сделать это может только пользователь с правами администратора в Конфигуратор SePlatform.Security. Минимальный срок действия пароля указан в праве **Срок действия пароля** (**PasswordAge**).

1.5. Собственный конфигуратор подсистемы безопасности

В разделе продемонстрировано создание собственного конфигуратора подсистемы безопасности SePlatform.Security.

Описано, как с помощью компонентов расширения SePlatform.HMI.Security:

- получать данные об учетных записях пользователей, группах пользователей и приложениях, имеющихся в подсистеме безопасности SePlatform.Security, из проектов SePlatform.HMI;
- менять имеющиеся учетные записи, группы и приложения из проектов SePlatform.HMI;
- создавать новые учетные записи, группы и приложения из проектов SePlatform.HMI;
- сохранять резервные копии конфигурации SePlatform.Security и восстанавливать их.

ПРИМЕЧАНИЕ

Раздел описывает расширенный сценарий применения компонентов SePlatform.HMI.Security. В большинстве случаев создавать собственный конфигуратор не обязательно: для конфигурирования подсистемы безопасности достаточно использовать Конфигуратор SePlatform.Security.

Описанный проект-пример представляет собой простейший конфигуратор подсистемы безопасности.

1. Скачайте архив с проектом-примером `OwnConfigurator.hmi` и откройте в Дизайнер SePlatform.HMI.
2. Запустите Конфигуратор SePlatform.Security для просмотра конфигурации подсистемы безопасности.

ПРИМЕЧАНИЕ

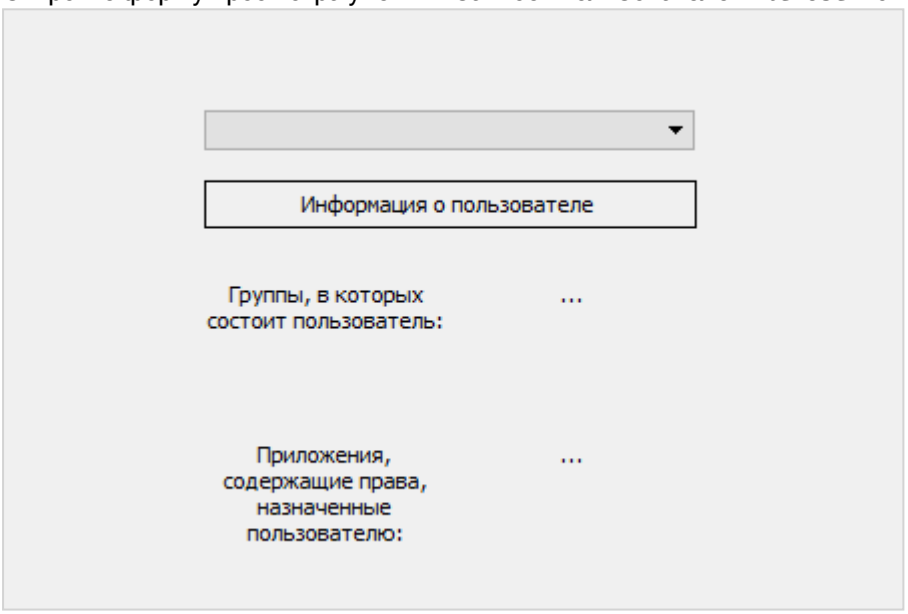
Проект-пример позволяет создать новые и просматривать имеющиеся учетные записи пользователей. Создание и просмотр групп пользователей и приложений можно реализовать с помощью аналогичных компонентов, их свойств, функций и событий, указанных в примечании к каждому подразделу.

Просмотр учетных записей пользователей

ОБРАТИТЕ ВНИМАНИЕ

Просмотр и изменение текущей конфигурации подсистемы безопасности возможен только для пользователя с правами администратора.

Откройте форму просмотра учетных записей пользователей *ViewUserForm* в проекте-примере.



На форму *ViewUserForm* добавлены:

- *SecurityManager* - элемент типа **Настройка безопасности: Менеджер** (стр. 76), позволяющий запрашивать и удалять имеющиеся учетные записи, группы и приложения;
- *SecurityManagerUser* - элемент типа **Настройка безопасности: Пользователь** (стр. 97), позволяющий добавлять новые и менять имеющиеся учетные записи пользователей.

Извлечение списка учетных записей из подсистемы безопасности

Для извлечения списка учетных записей используются возможности элемента *SecurityManager*:

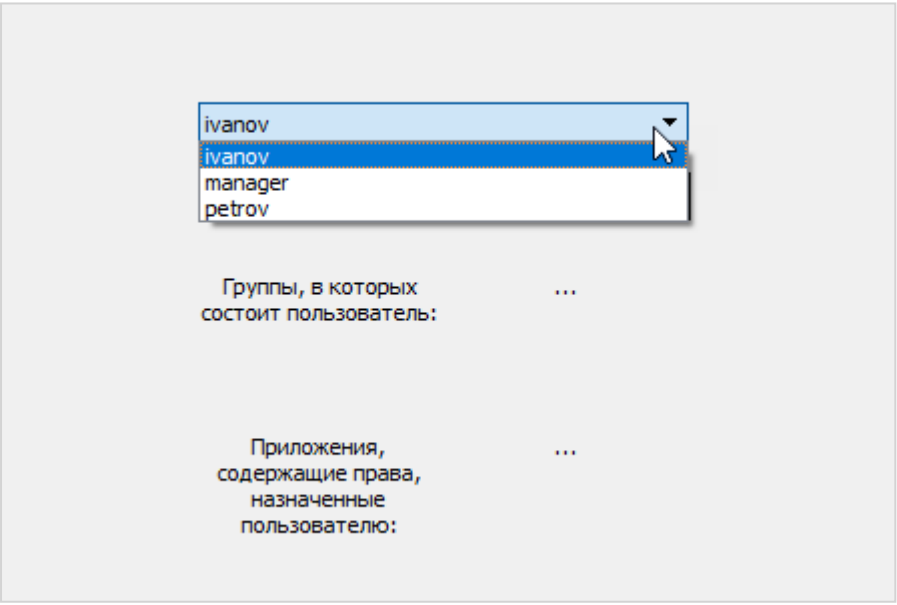
1. При запуске окна формы *ViewUserForm* вызывается функция **RequestUsersList()** (стр. 78), запрашивающая список учетных записей из подсистемы безопасности SePlatform.Security.
2. Если список получен, активируется событие **RequestUsersListComplete** (стр. 82).

Результат активации события RequestUsersListComplete - список учетных записей пользователей - хранится в переменной JSONUsersList. Данные хранятся в формате JSON. Ознакомьтесь с кодом обработчика события RequestUsersListComplete элемента *SecurityManager* из проекта-примера, написанном на языке JavaScript.

```
var usrs = JSON.parse(JSONUsersList);
//преобразование данных из формата JSON в объект
for (var i = 0; i < usrs.data.length; i++)
{
 //для каждого массива внутри объекта usrs
 unit.Global.UserName[i] = usrs.data[i].symbolicId;
 //извлечение имени пользователя и помещение его в глобальную переменную
 unit.Global.UserId[i] = usrs.data[i].uid;
 //извлечение uid пользователя и помещение его в глобальную переменную
 ComboBox_Users.AddItem(unit.Global.UserName[i]);
 //добавление имени пользователя в выпадающий список
}
```

Данный скрипт извлекает из JSON-структуры логины (symbolicId) и их идентификаторы (uid). Для удобства дальнейшего использования извлеченные данные помещаются в глобальные переменные *UserName[]* и *UserId[]* соответственно.

Запустите проект-пример в рантайм. Откроется окно выбора действия.
Нажмите кнопку **Посмотреть информацию о пользователях**. Откроется окно просмотра информации об учетных записях пользователей. В выпадающий список попадут логины всех учетных записей из подсистемы безопасности.



ПРИМЕЧАНИЕ

Аналогично из подсистемы безопасности SePlatform.Security извлекаются списки групп и приложений:

- для получения списка групп используйте функцию RequestGroupList() (стр. 77), событие RequestGroupListComplete (стр. 81) и переменную JSONGroupList элемента *SecurityManager*;
- для получения списка приложений используйте функцию RequestAppList() (стр. 77), событие RequestAppListComplete (стр. 81) и переменную JSONAppList элемента *SecurityManager*.

Просмотр информации о пользователе

Для того, чтобы просмотреть информацию о выбранной из выпадающего списка учетной записи, её необходимо сначала загрузить в элемент *SecurityManagerUser*. Это можно сделать с помощью функции **Load()** (стр. 99), входным параметром для которой является уникальный идентификатор выбранной учетной записи (uid). Ознакомьтесь с кодом обработчика события **MouseClick** кнопки **Информация о пользователе**.

```
SecurityManagerUser.Load(unit.Global.UserId[ComboBox_Users.SelectedIndex]);
//загрузка информации о выбранной учетной записи в объект SecurityManagerUser по
идентификатору
Groups_List.Text = "";
//очистка информационного поля при выборе новой УЗ
Apps_List.Text = "";
//очистка информационного поля при выборе новой УЗ
```

После загрузки данных активируется событие **LoadComplete** (стр. 107) элемента *SecurityManagerUser*. Ознакомьтесь с кодом обработчика этого события, написанном на языке JavaScript.

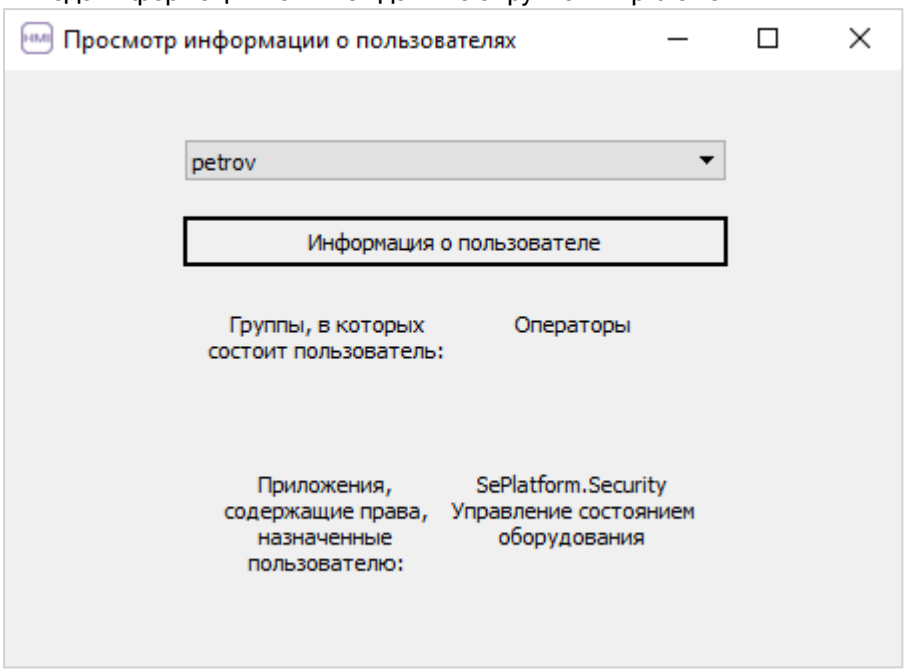
```
JSONGroupList = SecurityManagerUser.GetGroupsList();
//получение списка групп, в которых состоит пользователь
JSONAppList = SecurityManagerUser.GetApplicationsList();
//получения списка приложений, содержащих права пользователя
var grpslst = JSON.parse(JSONGroupList);
var applst = JSON.parse(JSONAppList);
for (var v of grpslst.data)
//для всех элементов коллекции
{
 Groups_List.Text += v.groupName + "\n";
 //добавление названия группы в список
}
for (var v of applst.data)
//для всех элементов коллекции
{
 Apps_List.Text += v.appName + "\n";
 //добавление названия приложения в список
}
```

Данные об учетной записи извлекаются из элемента *SecurityManagerUser* путем обращения к его свойствам и функциям. Например:

- функция **GetGroupsList()** (стр. 101) возвращает данные о группах, в которых состоит пользователь, в JSON-формате;
- функция **GetApplicationsList()** (стр. 101) возвращает данные о приложениях с правами, назначенными пользователю, в JSON-формате.

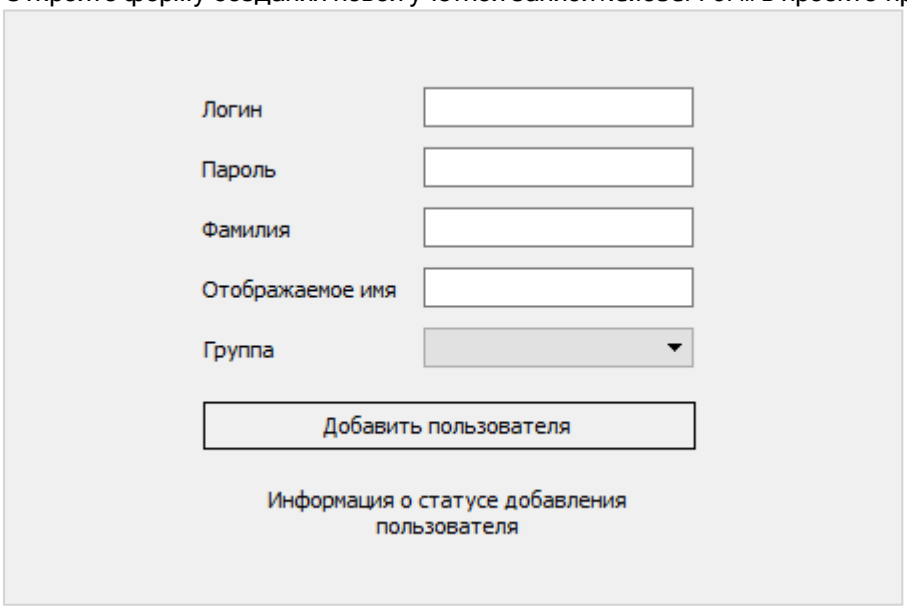
Поместив данные в переменные *JSONGroupList* и *JSONAppList*, можно извлечь из JSON-структуры необходимую информацию.

Запустите проект-пример в рантайм. Откроется окно выбора действия. Нажмите кнопку **Посмотреть информацию о пользователях**. Откроется окно просмотра информации об учетных записях пользователей. Выберите любой логин из выпадающего списка и нажмите кнопку **Информация о пользователе**. В поле вывода информации появятся данные о группах и приложениях.



Создание новой учетной записи пользователя

Откройте форму создания новой учетной записи *NewUserForm* в проекте-примере.



На форму создания новой учетной записи *NewUserForm* добавлены:

- *SecurityManager* - элемент типа **Настройка безопасности: Менеджер** (стр. 76), позволяющий запрашивать и удалять имеющиеся учетные записи, группы и приложения;
- *SecurityManagerUser* - элемент типа **Настройка безопасности: Пользователь** (стр. 97), позволяющий добавлять новые и менять существующие учетные записи пользователей.

Для добавления новой учетной записи используйте функции и свойства элемента *SecurityManagerUser*. Ознакомьтесь с кодом обработчика события MouseClick кнопки **Добавить пользователя**, написанном на языке SePlatform.Om.

```
SecurityManagerUser.New();
//создание новой УЗ пользователя
SecurityManagerUser.Login=TextEdit_Login;
//присвоение логина новой УЗ
SecurityManagerUser.SetPassword(TextEdit_Pswd);
//присвоение пароля новой УЗ
SecurityManagerUser.Surname=TextEdit_Surname;
//присвоение фамилии новой УЗ
SecurityManagerUser.DispalyName=TextEdit_DisplayName;
//присвоение отображаемого имени новой УЗ
SecurityManagerUser.AddGroup(unit.Global.GroupId[ComboBox_Group.SelectedIndex]);
//вызов функции AddGroup для добавления создаваемого пользователя в выбранную в выпадающем
списке группу
SecurityManagerUser.Save();
//сохранение введенных данных
```

Здесь:

- функция **New()** (стр. 104) создает, а функция **Save()** (стр. 99) - сохраняет новую учетную запись в подсистеме безопасности;
- свойства элемента *SecurityManager* (**Login**, **Surname**, **DisplayName** (стр. 97)) получают значения, введенные в текстовые поля;
- функция **SetPassword()** (стр. 100) принимает введенное значение в качестве пароля;
- функция **AddGroup()** (стр. 105) принимает uid группы, выбранной из выпадающего списка, в качестве входного параметра и добавляет пользователя в соответствующую группу.

ПРИМЕЧАНИЕ

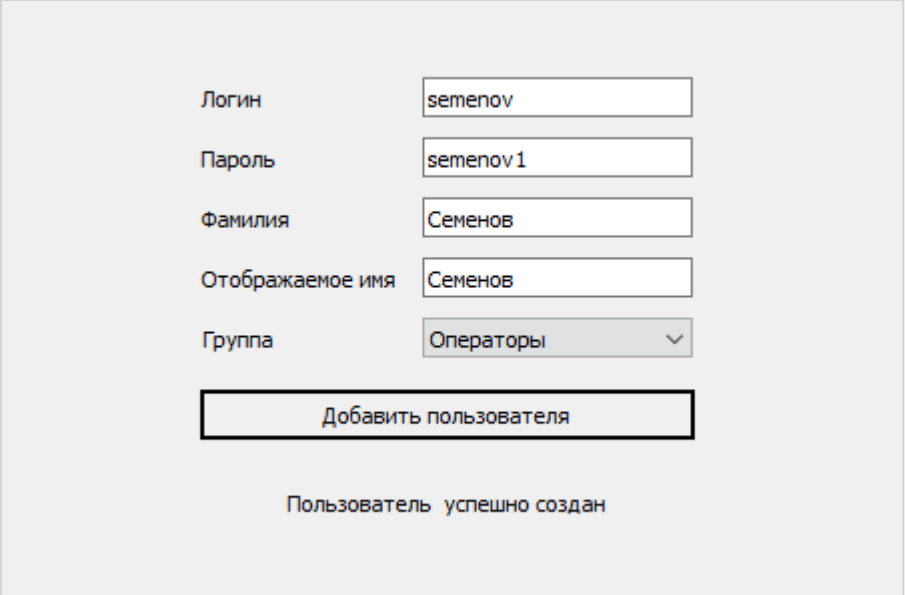
Пользователю можно назначить только одну группу.

ВАЖНО

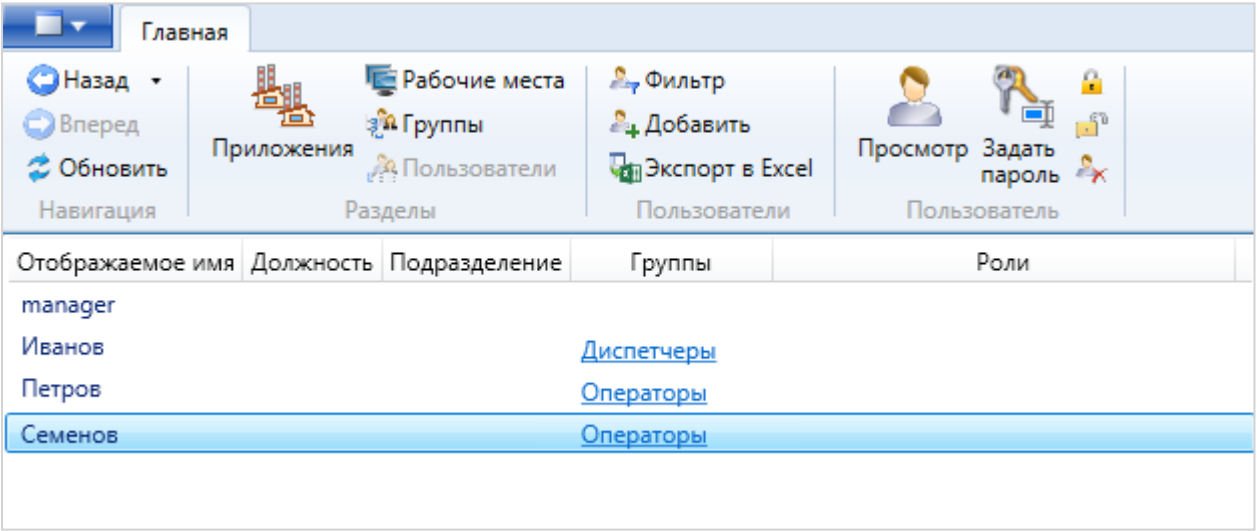
Будьте внимательны при работе с проектом-примером. Описанные ниже действия ведут к изменению конфигурации SePlatform.Security.

Запустите проект-пример в рантайм. Откроется окно выбора действия.

Нажмите кнопку **Создать нового пользователя**. Откроется окно создания новой учетной записи. Попробуйте ввести данные новой учетной записи и нажать кнопку **Добавить пользователя**. Система вернет сообщение об успешности операции.



В Конфигуратор SePlatform.Security нажмите кнопку **Обновить** и убедитесь в том, что новая учетная запись создана.



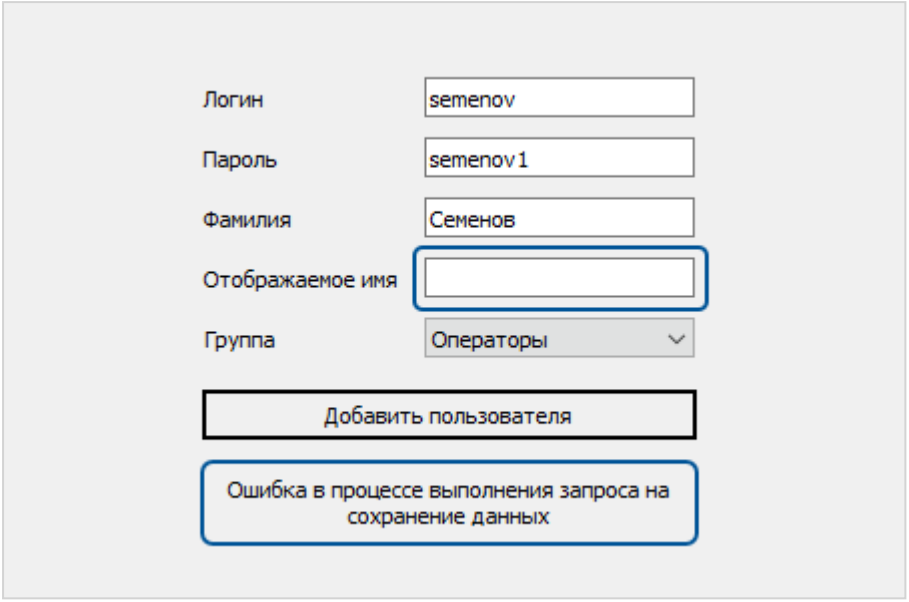
ОБРАТИТЕ ВНИМАНИЕ

Набор полей ввода и необходимость их заполнения можно определять в ходе разработки собственного конфигуратора. Для подсистемы безопасности SePlatform.Security обязательными являются:

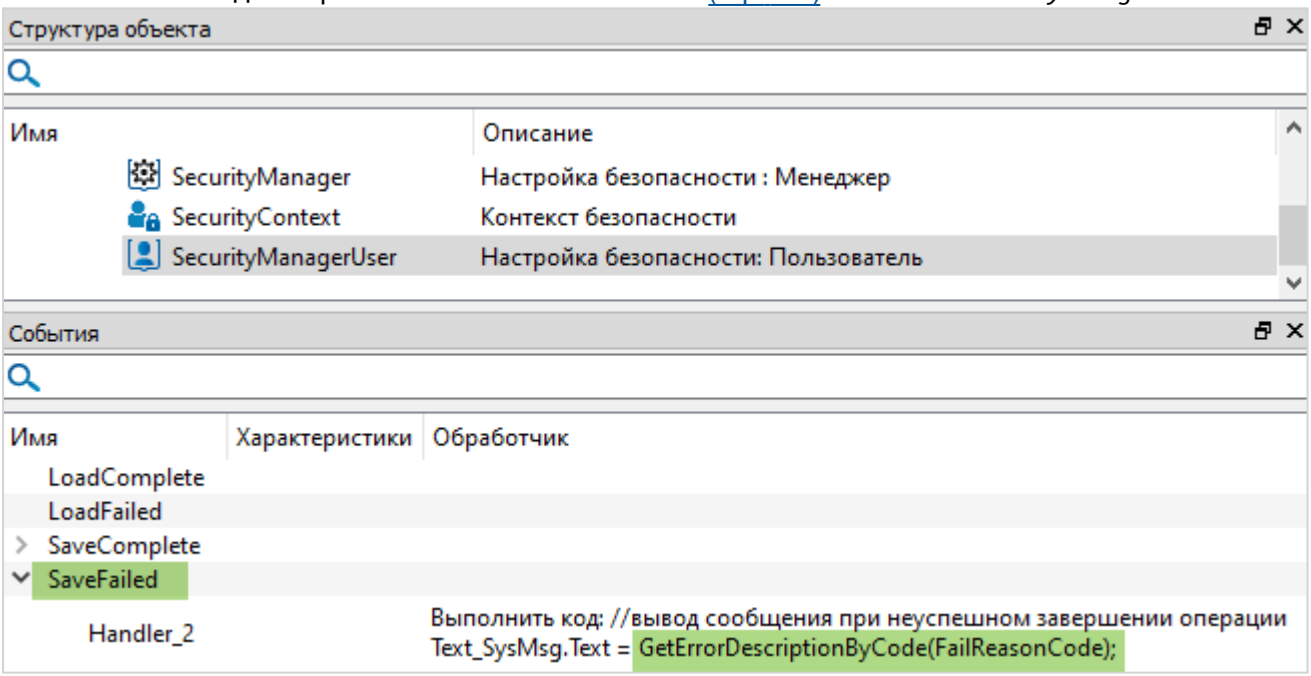
- значение свойства Login;
- значение свойства Surname;
- значение свойства DisplayName;
- результат вызова функции SetPassword().

Остальные свойства, функции и ограничения на их значения применяются исходя из требований к разрабатываемому конфигуратору.

Если бы при создании новой учетной записи не было заполнено хотя бы одно из обязательных для подсистемы безопасности полей, сохранить учетные данные не удалось бы.

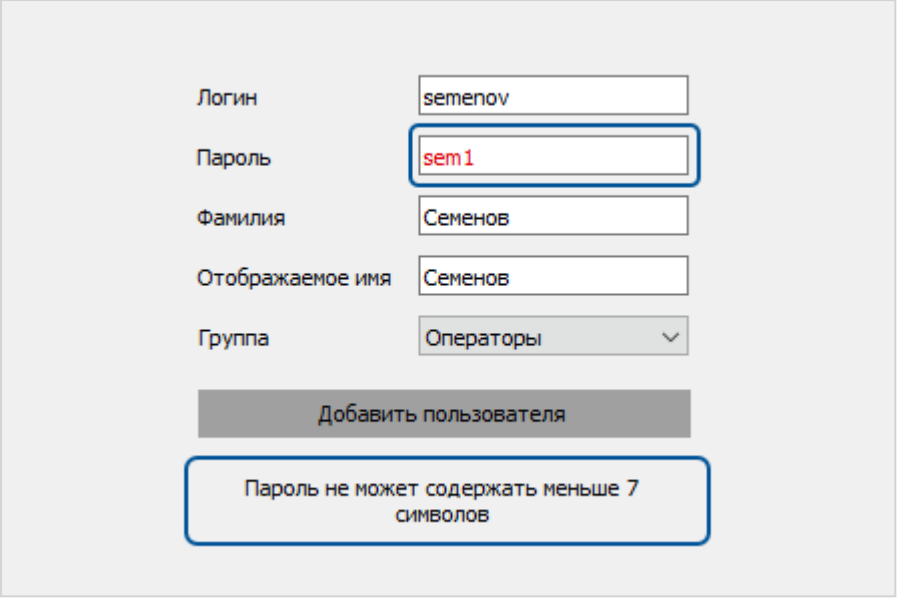


Ознакомьтесь с кодом обработчика события SaveFailed (стр. 107) элемента *SecurityManagerUser*.



Подсистема безопасности не принимает новую учетную запись без заполнения обязательного поля. Текст ошибки можно получить с помощью функции **GetErrorDescriptionByCode()** (стр. 107), входным параметром для которой является переменная FailReasonCode, хранящая код ошибки.

Попробуйте ввести пароль, не содержащий цифр и/или состоящий менее чем из семи символов.



Проверка требований к паролю настроена при разработке проекта-примера и выполняется в обработчике события **TextChanged** элемента *TextEdit_Pswd*. Сама подсистема безопасности не накладывает требований к содержанию пароля создаваемой учетной записи.

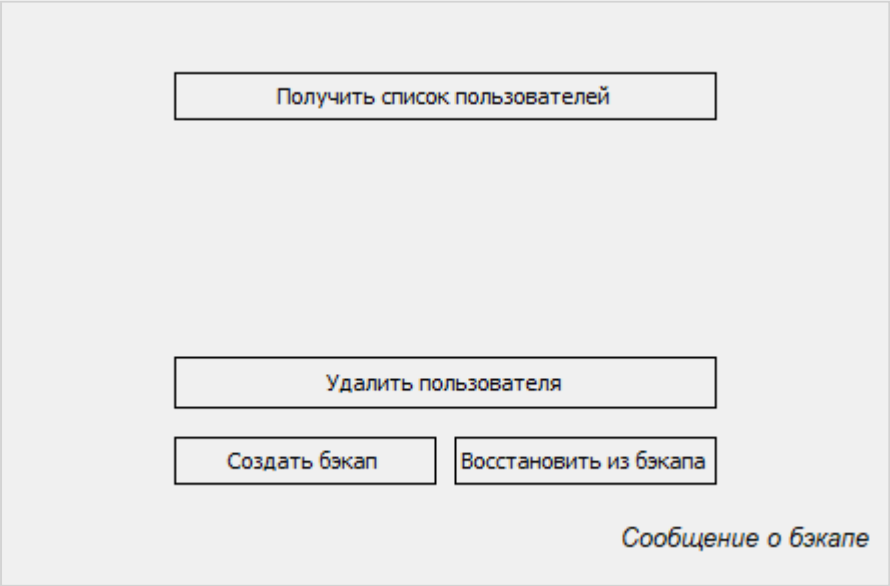
ПРИМЕЧАНИЕ

Аналогично в подсистеме безопасности SePlatform.Security создаются группы и приложения:

- для создания групп пользователей используйте функции и свойства элемента типа **Настройка безопасности: Группа** (стр. 108);
- для создания приложений используйте функции и свойства элемента типа **Настройка безопасности: Приложение** (стр. 86).

Создание и восстановление резервной копии

Откройте форму управления резервными копиями *Backups* в проекте-примере.



На форму *Backups* добавлен *SecurityManager* - элемент типа **Настройка безопасности: Менеджер** (стр. 76).

Создание резервной копии

Для создания резервных копий используются возможности элемента *SecurityManager*:

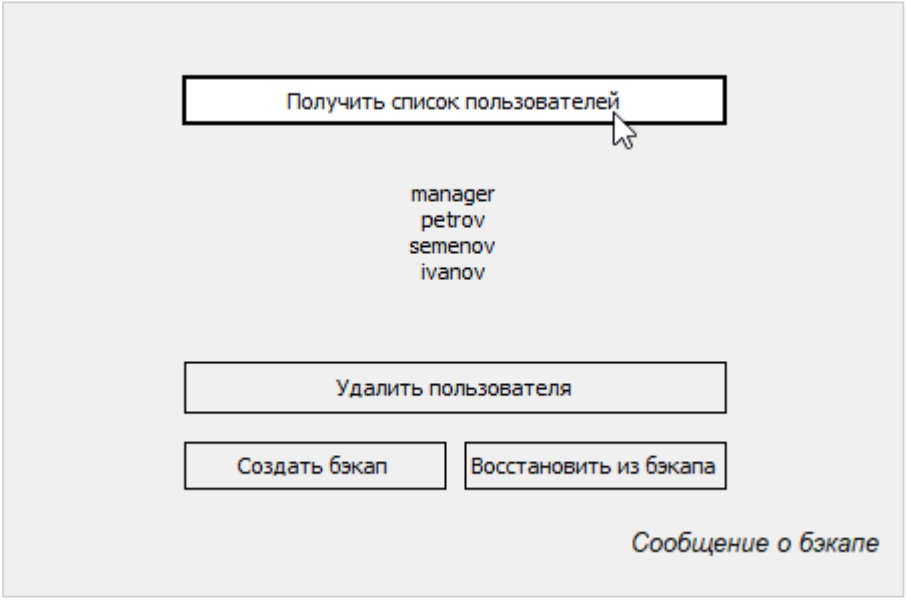
1. При нажатии кнопки *Button_CreateBckp* вызывается функция **ExportConfiguration()** (стр. 80), создающая файл резервной копии в указанном расположении. В данном случае резервная копия создается в папке текущего пользователя ОС. Ознакомьтесь с кодом обработчика **MouseClick** кнопки *Button_CreateBckp*:

```
SecurityManager.ExportConfiguration(FileSystem.CurrentUserFolder + "/export.ldb")
```

2. После выполнения функции активируется одно из событий *SecurityManager*: в случае успешного создания резервной копии - **GetConfigurationFinished** (стр. 85), в противном случае - **GetConfigurationFailed** (стр. 86). Ознакомьтесь с обработчиками этих событий.

> ВАЖНО
> Будьте внимательны при работе с проектом-примером. Описанные ниже действия ведут к изменению конфигурации SePlatform.Security.

Запустите проект-пример в рантайм. Откроется окно выбора действия.
Нажмите кнопку **Управление резервными копиями**. Откроется окно формы *Backups*. Получите список пользователей, нажав на соответствующую кнопку.



Нажмите кнопку **Создать бэкап**. Будет вызвана функция **ExportConfiguration()**:

- в случае успешного выполнения функции создастся резервная копия и появится соответствующее сообщение;
- в случае неуспешного выполнения функции появится сообщение с текстом ошибки.

Восстановление резервной копии

Для восстановления резервных копий используются возможности элемента *SecurityManager*:

1. При нажатии кнопки *Button_OpenBckp* вызывается функция **ImportConfiguration()** (стр. 80), восстанавливающая резервную копию из указанного файла. В данном случае резервная копия восстанавливается из файла, расположенного в папке текущего пользователя ОС. Ознакомьтесь с кодом обработчика **MouseClick** кнопки *Button_OpenBckp*:

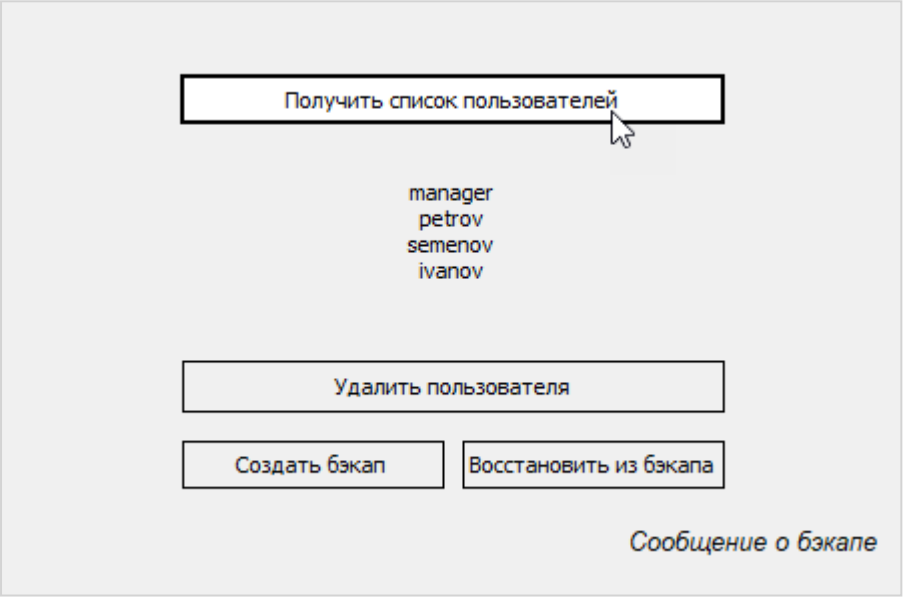
```
SecurityManager.ImportConfiguration(FileSystem.CurrentUserFolder + "/export.ldb")
```

2. После выполнения функции активируется одно из событий *SecurityManager*: в случае успешного восстановления конфигурации - **SetConfigurationFinished** (стр. 86), в противном случае - **SetConfigurationFailed** (стр. 86). Ознакомьтесь с обработчиками этих событий.

ВАЖНО

Будьте внимательны при работе с проектом-примером. Описанные ниже действия ведут к изменению конфигурации SePlatform.Security.

Запустите проект-пример в рантайм. Откроется окно выбора действия.
Нажмите кнопку **Управление резервными копиями**. Откроется окно формы *Backups*. Получите список пользователей, нажав на соответствующую кнопку.



ПРИМЕЧАНИЕ

Для наглядности можно удалить одного из пользователей. Для этого нажмите **Удалить пользователя** и в открывшемся окне удалите любую учетную запись, кроме записи администратора. Затем вернитесь к окну управления резервными копиями. Получите список пользователей, нажав соответствую кнопку. Убедитесь, что учетная запись была удалена.

Нажмите кнопку **Восстановить из бэкапа**. Будет вызвана функция ImportConfiguration():

- в случае успешного выполнения функции восстановится резервная копия из файла;
- в случае неуспешного выполнения функции появится сообщение с текстом ошибки.

ПРИМЕЧАНИЕ

Вновь получите список пользователей, нажав соответствую кнопку. Убедитесь, что удаленная учетная запись восстановлена.

1.6. Контроль целостности

В разделе описано как получать информацию об изменениях в файлах и папках с помощью компонентов расширения SePlatform.HMI.Security. Для этого представлен проект-пример, контролирующий целостность указанной папки и её содержимого.

ПРИМЕЧАНИЕ

Раздел описывает расширенный сценарий применения компонентов SePlatform.HMI.Security. Показано, как создавать эталон состояния файлов, запускать проверку целостности и использовать результат проверки в проектах SePlatform.HMI.

Для ознакомления с этим разделом:

1. Скачайте архив с проектом-примером `IntegrityControl.hmi` и откройте в Дизайнер SePlatform.HMI. В архив также включена папка `controlled objects`. В проекте-примере будет проверяться состояние этой папки и входящих в нее файлов.

2. Перейдите к настройками подсистемы безопасности SePlatform.Security. В конфигурационном файле `seplatform.security.agent.xml`, расположенном в `C:\Program Files\SePlatform\SePlatform.Security` (для Windows) или в `/opt/SePlatform/SePlatform.Security` (для Linux), включите режим контроля целостности. Для этого назначьте атрибуту `ICMode` тега `<Options>` значение равное единице.

```
<Options LoggerLevel="2" ICMode="1" ... />
```

3. В конфигурационном файле `seplatform.security.ic.xml`, расположенном здесь же, укажите контролируемую папку с вложенными в нее файлами, задайте таймер проверки указанных файлов и определите серьезность нарушений целостности файлов.

   3.1. Чтобы указать контролируемую папку, назначьте атрибуту `IC file` тега `<ICList>` значение, являющееся полным путем к папке `controlled objects` из скачанного архива.

```
<SePlatform.Integrity.Control>
	<ICList>
		<IC file="D:\controlled objects"/>
	</ICList>
</SePlatform.Integrity.Control>
```

   3.2. Чтобы задать таймер проверки указанных файлов, назначьте атрибуту `ICPeriodSeconds` тега `<Options>` значение интервала проверок, например, равным 300 секунд.

```
<SePlatform.Integrity.Control>
	<ICList>
		<IC file="D:\controlled objects"/>
	</ICList>
	<Options ICPeriodSeconds="300"/>
</SePlatform.Integrity.Control>
```

   Если необходимо, чтобы проверка проводилась только по запросу пользователя, укажите атрибуту `ICPeriodSeconds` значение **«0»**.

3.3. Чтобы определить серьезность нарушений целостности файлов, добавьте в тег `<Options>` новые атрибуты и назначьте им значения типа `bool`. Новыми атрибутами могут быть:

- атрибут `ICErrorObjectNotExist`, определяющий серьёзность нарушения "Объект не существует".
- атрибут `ICErrorDateChanged`, определяющий серьёзность нарушения "Изменилась дата объекта".
- атрибут `ICErrorFileChanged`, определяющий серьёзность нарушения "Изменилось содержимое файла".
- атрибут `ICErrorNewObject`, определяющий серьёзность нарушения "Обнаружен новый объект".

Если указать новому атрибуту значение `true`, то нарушение данного типа будет считаться ошибкой, если `false` - нарушение данного типа будет считаться предупреждением. По умолчанию все нарушения считаются ошибками.

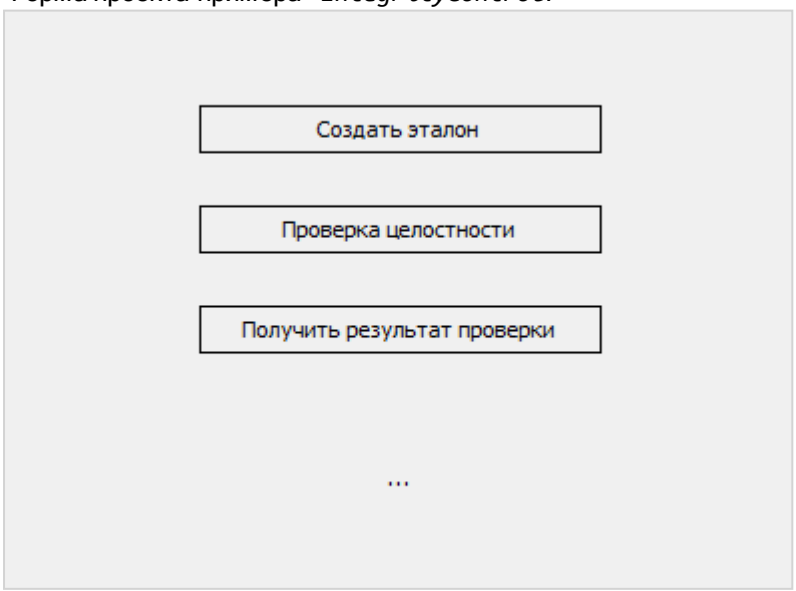
ПРИМЕЧАНИЕ

Вложенные в контролируемую папку файлы и папки можно исключать из контроля целостности. Для этого укажите полный путь к исключаемым файлам и папкам здесь же в качестве значения атрибута `IC file` тега `<ICExclude>`.

```
<SePlatform.Integrity.Control>
	<ICList>
		<IC file="D:\controlled objects"/>
	</ICList>
	<ICExclude>
		<IC file="D:\controlled objects\3.txt"/>
	</ICExclude>
	<Options ICPeriodSeconds="300"/>
</SePlatform.Integrity.Control>
```

Проект-пример

Форма проекта-примера - *IntegrityControl*:

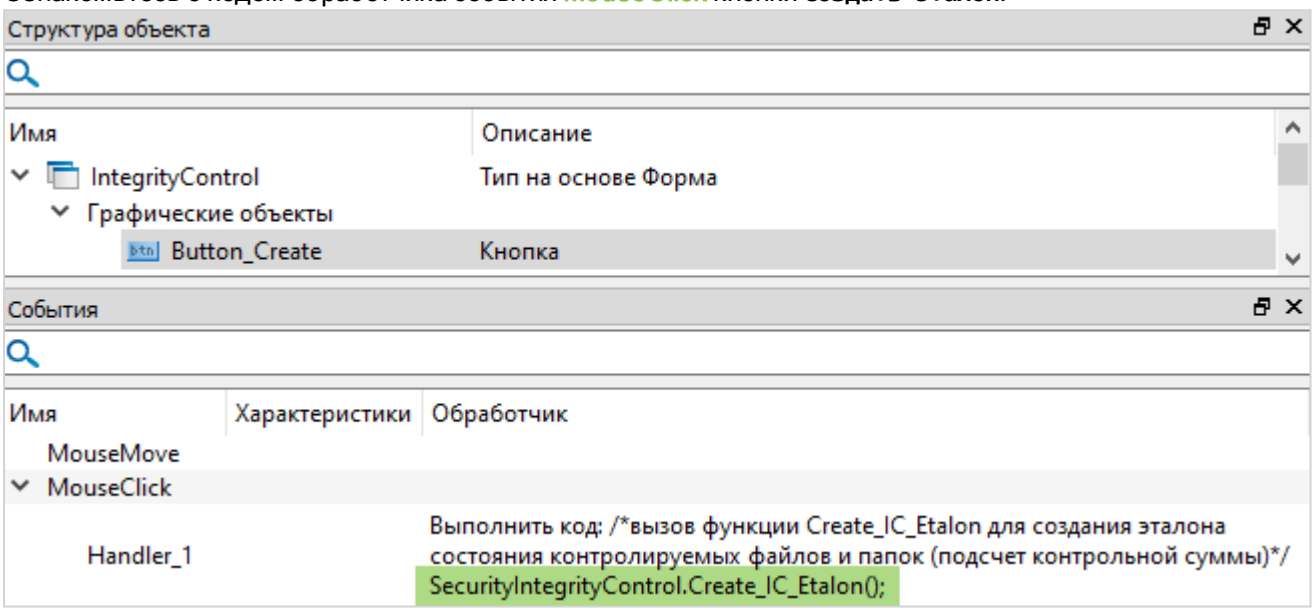


На форму добавлены:

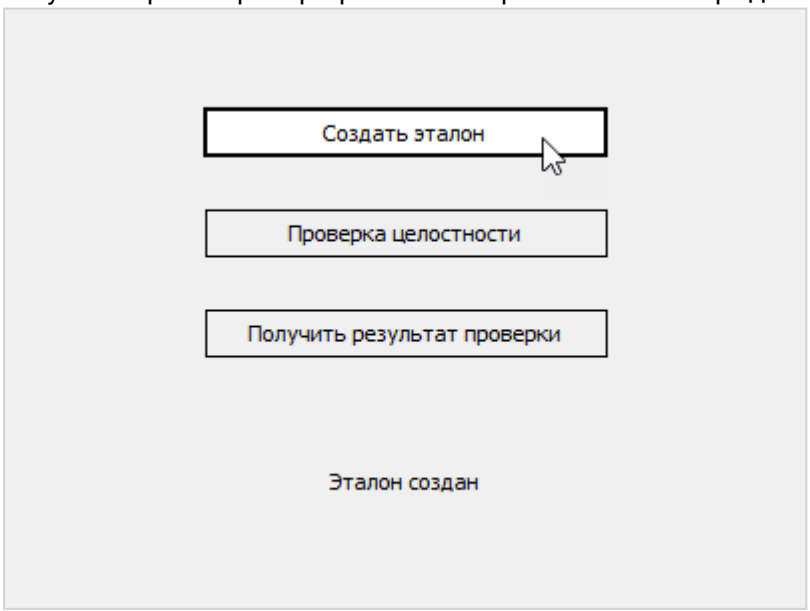
- *SecurityIntegrityControl* - элемент типа **Настройка безопасности: Контроль целостности** (стр. 64), позволяющий управлять проверками контролируемых файлов;
- *SecurityContext* - элемент типа **Контекст безопасности** (стр. 47), обеспечивающий взаимодействие с SePlatform.Security.

Создание эталона состояния файлов

Для проверки текущего состояния контролируемых файлов необходимо сверять их с версией, которая отвечает требованиям разрабатываемого проекта. Такая версия называется эталоном. Для того, чтобы создать эталон, необходимо привести файлы в нужное состояние, а затем создать эталон на основе текущего состояния файлов.
Ознакомьтесь с кодом обработчика события MouseClick кнопки **Создать эталон**:



При вызове функции Create_IC_Etalon() (стр. 67) элемента *SecurityIntegrityControl* формируется эталон состояния контролируемых файлов. Он хранится в виде зашифрованного файла `seplatform.security.ic_etalon.xml`.
Запустите проект-пример в рантайм. Откроется окно выбора действия. Нажмите кнопку **Создать эталон**:



При нажатии кнопки **Создать эталон**:

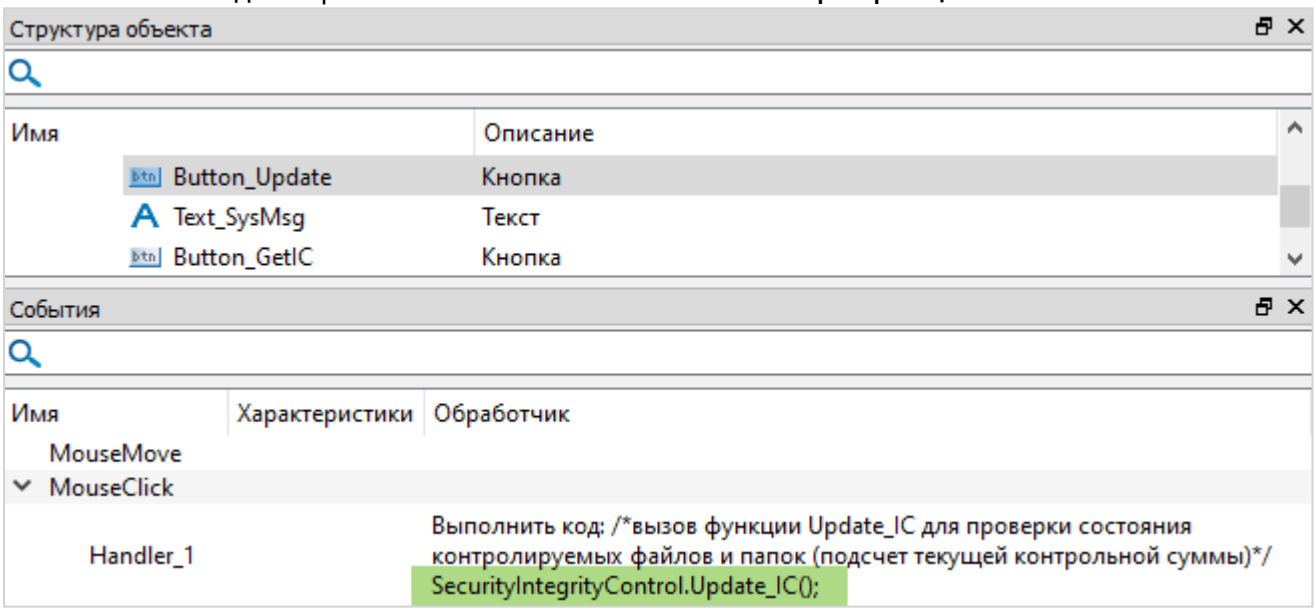
1. Была вызвана функция Create_IC_Etalon(), подсчитавшая контрольную сумму файлов и записавшая её в файл `seplatform.security.ic_etalon.xml`.
2. Активировалось событие CreateFinished (стр. 68) элемента *SecurityIntegrityControl*, в результате чего появилось сообщение **«Эталон создан»**.

ПРИМЕЧАНИЕ

Событие CreateFailed (стр. 69) элемента *SecurityIntegrityControl* позволяет обрабатывать ошибки процесса создания эталона. Подсистема безопасности возвращает текст ошибки в переменную errorMessage.

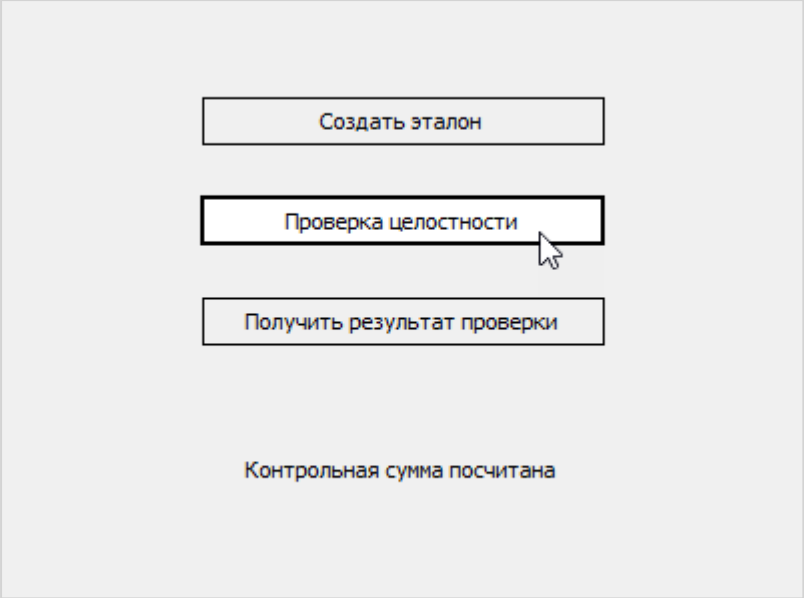
Проверка текущего состояния файлов

Проверка текущего состояния файлов представляет собой сравнение текущей контрольной суммы с эталонной.
Ознакомьтесь с кодом обработчика события MouseClick кнопки **Проверка целостности**:



При вызове функции Update_IC() (стр. 66) элемента *SecurityIntegrityControl* происходит сравнение текущего состояния файлов с эталоном.

Запустите проект-пример в рантайм. Откроется окно выбора действия. Нажмите кнопку **Проверка целостности**:



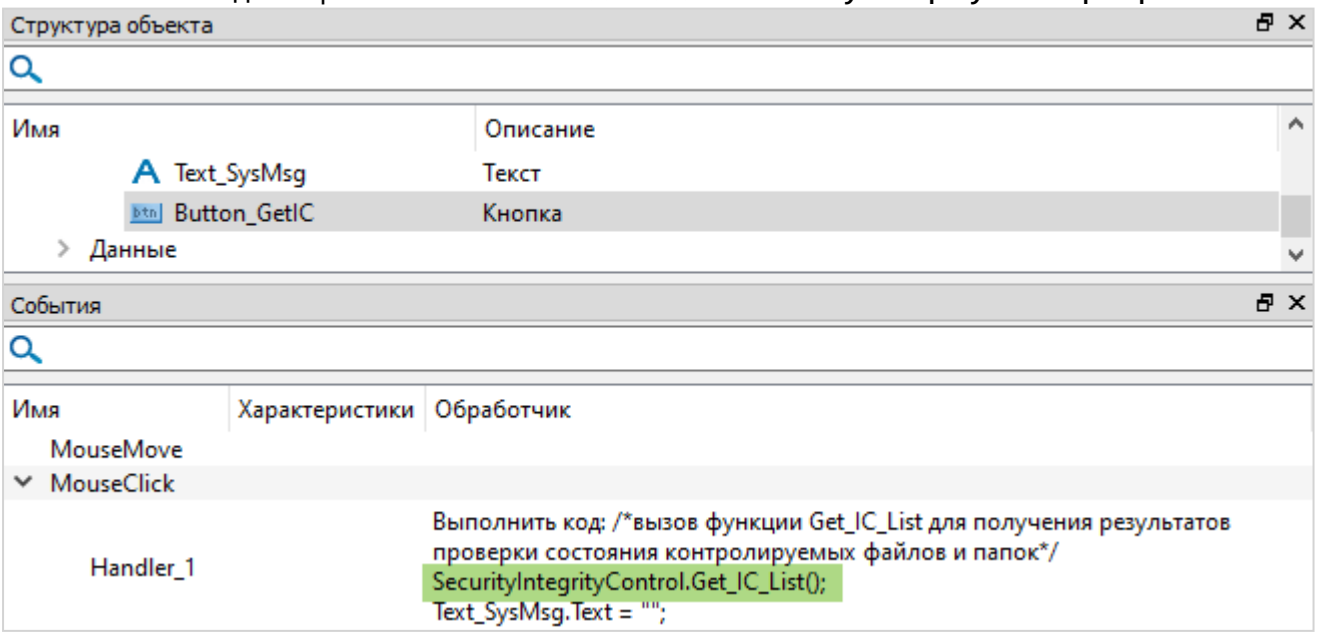
При нажатии кнопки **Проверка целостности**:

1. Была вызвана функция Update_IC(), подсчитавшая контрольную сумму файлов и сравнившая её с эталоном.
2. Активировалось событие UpdateFinished (стр. 68) элемента *SecurityIntegrityControl*, в результате чего появилось сообщение о том, что контрольная сумма посчитана.

> ПРИМЕЧАНИЕ
>
> Событие UpdateFailed (стр. 68) элемента *SecurityIntegrityControl* позволяет обрабатывать ошибки процесса подсчета контрольной суммы и сравнения с эталоном. Подсистема безопасности возвращает текст ошибки в переменную errorMessage.

Получение текущего состояния файлов

Для получения результата проверки целостности файлов, его необходимо вызвать.
Ознакомьтесь с кодом обработчика события MouseClick кнопки **Получить результат проверки**:



При вызове функции **Get_IC_List()** (стр. 66) элемента *SecurityIntegrityControl* происходит получение результата проверки целостности. В случае успешного завершения операции активируется событие **ListIsReady** (стр. 69) элемента *SecurityIntegrityControl*. Ознакомьтесь с кодом обработчика данного события, написанном на языке JavaScript.

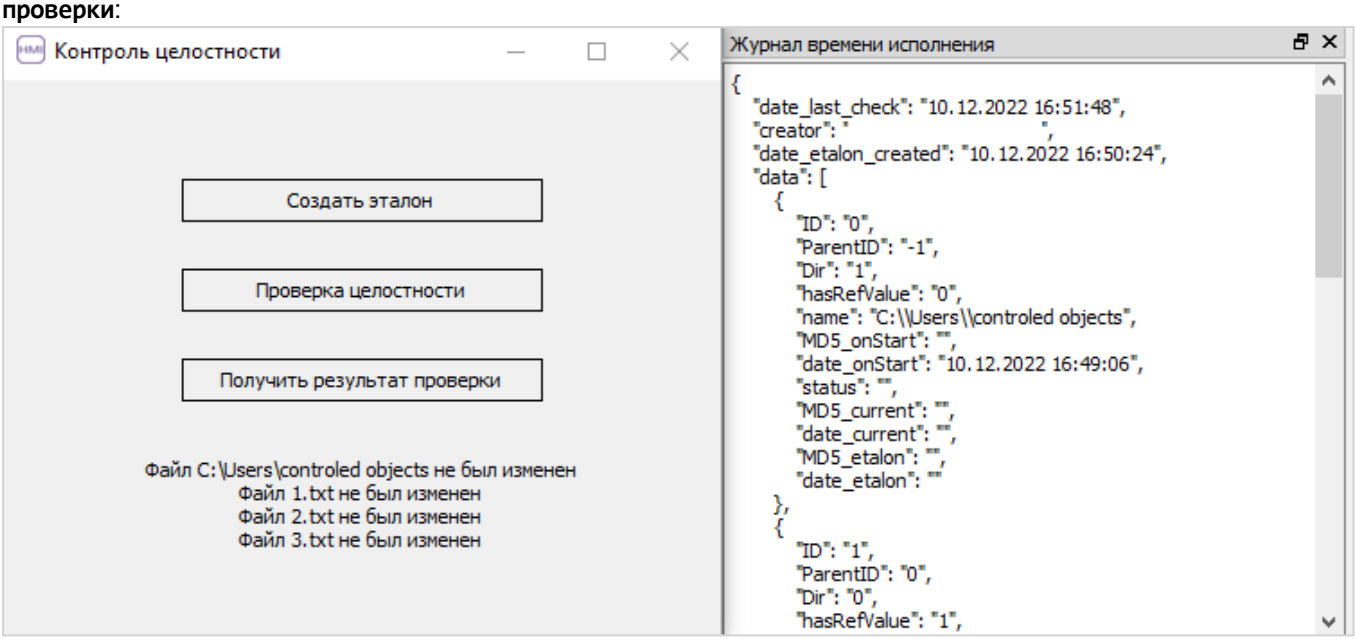
```
DebugTool.Log(IC_List_JSON);
//помещение в лог полного результата проверки целостности файлов
var IClst = JSON.parse(IC_List_JSON);
//преобразование данных из формата JSON в объект
for (var i = 0; i < 10; i++)
{
 if (IClst.data[i] != undefined)
 {
  //проверка параметра status, хранящего состояние файла
  if (IClst.data[i].status == 0)
  //проверка состояния файла: файл не был изменен
  {
   Text_SysMsg.Text += "Файл "+IClst.data[i].name+" не был изменен"+"\n";
  }
  if (IClst.data[i].status == 8)
  //проверка состояния файла: не была ли нарушена целостность
  {
   Text_SysMsg.Text += "Файл "+IClst.data[i].name+" был изменен"+"\n";
  }
  if (IClst.data[i].status == 4)
  //проверка наличия файла
  {
   Text_SysMsg.Text += "Файл "+IClst.data[i].name+" был удален или
перемещен"+"\n";
  }
 }
}
```

Результат проверки хранится в переменной IC_List_JSON в форме JSON.

ПРИМЕЧАНИЕ

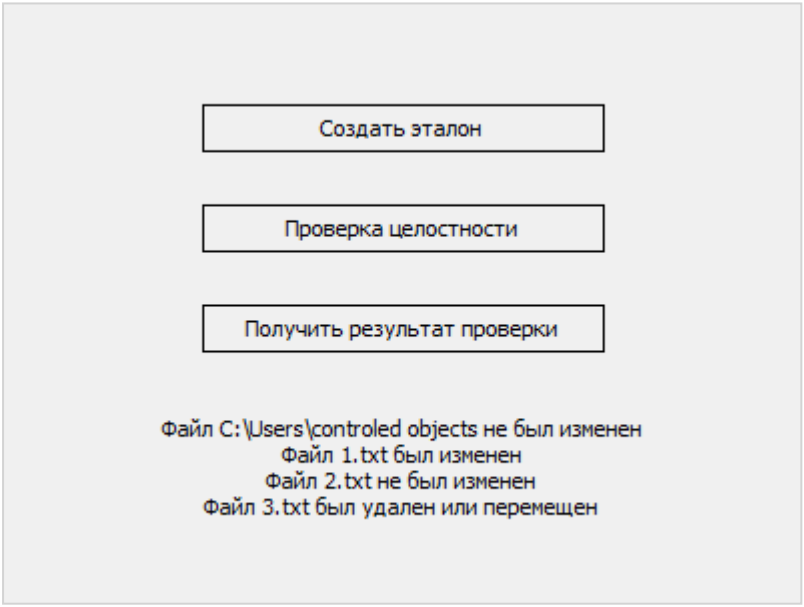
Событие **GetListFailed** (стр. 71) элемента *SecurityIntegrityControl* позволяет обрабатывать ошибки получения результата проверки целостности файлов. Подсистема безопасности возвращает текст ошибки в переменную errorMessage.

Запустите проект-пример в рантайм. Откроется окно выбора действия. Нажмите кнопку **Получить результат проверки**:



Файлы не были изменены, о чем говорится в сообщении. С результатом проверки в формате JSON можно ознакомиться в **Журнал времени исполнения**. Расшифровка результата приведена в описании свойства IC_List_JSON компонента Настройка безопасности: Контроль целостности (стр. 64).

Внесите изменения в контролируемые файлы. Например, напишите что-нибудь в файле `1.txt`, а файл `3.txt` удалите. Выполните проверку целостности и вызовите результат проверки:



Расшифровка сообщений о проверке целостности файлов

Сообщения о выполнении проверки целостности файлов можно получать в другие приложения. Например, в Service - LogViewer сообщения приходят автоматически после включения контроля целостности в настройках Агент SePlatform.Security. Сообщения имеют следующий вид:

```
Проверено XXX / YYY объектов. Обнаружено нарушений AA, ошибок A1, предупреждений A2. Скрыто эталоном BB.
```

Здесь:

- **«XXX»** - количество файлов, которые записаны в эталон. Это число постоянно, если эталон не пересоздавать.
- **«YYY»** - фактическое количество файлов в подконтрольных папках. Контроль целостности выполняется для всех этих файлов, из которых только XXX записаны в эталон.
- **«AA»** - общее количество нарушений. Среди них:
  - **«A1»** - количество нарушений, считающихся ошибками.
  - **«A2»** - количество нарушений, считающихся предупреждениями.

  Как определить тип нарушения описано в начале этого раздела, в пункте 3.3 настройки контроля целостности.
- **«BB»** - количество файлов, к которым не удалось получить доступ на момент создания эталона. Не входят в общую статистику ошибок.

1.7. Конфигурирование агента безопасности

Настройка службы **SePlatform.Security.Agent** (на ОС Windows) или **SePlatform.Security.service** (на ОС Linux) обычно выполняется в конфигурационном файле `seplatform.security.agent.xml`, расположенном в `C:\Program Files\SePlatform\SePlatform.Security` (для Windows) или в `/opt/SePlatform/SePlatform.Security` (для Linux). Подробнее назначение и настройка службы описаны в документе на SePlatform.Security, в разделах **«О продукте»** и **«Настройка компонентов»** соответственно.

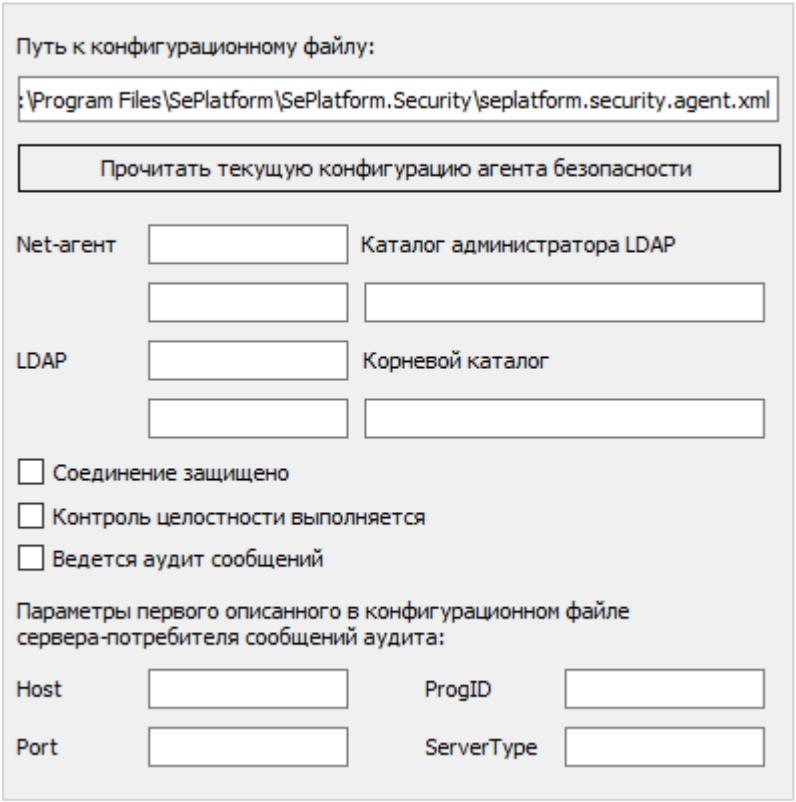
В этом разделе продемонстрирована возможность конфигурирования агента безопасности из проекта SePlatform.HMI с помощью компонента расширения SePlatform.HMI.Security. Для этого описан проект-пример, в котором можно ознакомиться с текущими настройками агента с помощью экземпляра компонента **Мастер конфигурирования Security**.

Для ознакомления с этим разделом:

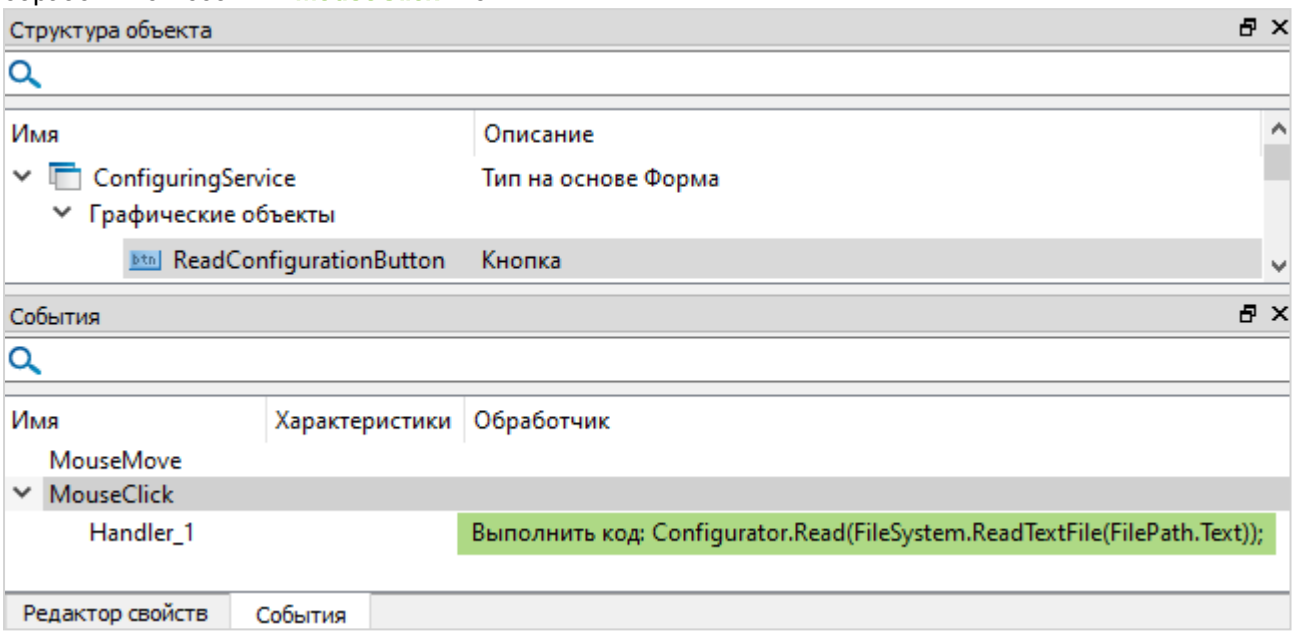
- Скачайте проект-пример и откройте его в Дизайнер SePlatform.HMI.
- Ознакомьтесь с настройками подсистемы безопасности SePlatform.Security: откройте конфигурационный файл `seplatform.security.agent.xml`, расположенный в `C:\Program Files\SePlatform\SePlatform.Security` (для Windows) или в `/opt/SePlatform/SePlatform.Security` (для Linux).

Состав проекта-примера

Единственная форма в проекте - *ConfiguringService*.



На форму добавлен элемент типа **Мастер конфигурирования Security** (стр. 116) - *Configurator*. Для получения текущих настроек агента безопасности сначала нужно вызвать функцию **Read()** (стр. 122) элемента. Функция вызывается по нажатию на кнопку *ReadConfigurationButton*. Ознакомьтесь с обработчиком события **MouseClick** кнопки:



При вызове функции выполняется обращение к конфигурационному файлу, путь к которому указан в качестве входного параметра функции. В данном случае путь к файлу хранится в текстовом поле *FilePath*, расположенном в верхней части окна проекта-примера. В случае успешного завершения операции:

› Значения из конфигурационного файла записываются в элемент *Configurator*. Чтобы обратиться к ним, используются свойства и функции компонента.

› Активируется событие **ReadingFinished** (стр. 134) элемента *Configurator*. Ознакомьтесь с кодом обработчика данного события:

```
//запись в лог информации об успешном выполнении операции
DebugTool.Log("Конфигурация прочитана");
//вызов команд, записывающих информацию о настройках агента
//в соответсвующих поля формы и в лог
WriteBasic.Invoke();
WriteAudit.Invoke();
```

Для удобства действия, выполняемые при наступлении события, разделены на две команды - *WriteBasic* и *WriteAudit*.

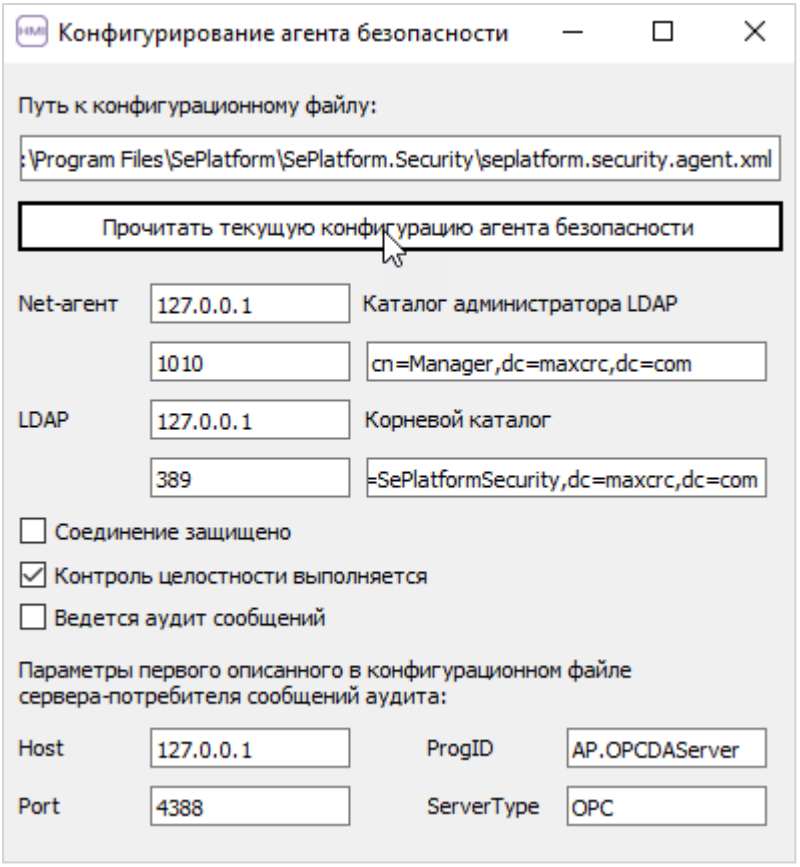
› Команда *WriteBasic* записывает в соответствующие поля основную информацию о настройках агента безопасности - параметры используемого Net-агента, LDAP-сервера, данные администратора и корневого каталога, сведения о ведении аудита, контроле целостности и защищенности соединения агента с LDAP-сервером. Ознакомьтесь с кодом обработчика события **Invoked** команды:

```
//запись значений свойств элемента Configurator в текстовые поля
Host.Text = Configurator.NetHost;
Port.Text = String.ToString(Configurator.NetPort);
LDAPHost.Text = Configurator.LdapHost;
SecureConnectionCheckbox.State = Configurator.UseSecureConnection;
ICModeCheckbox.State = Configurator.ICMode;
AuditCheckbox.State = Configurator.TraceAudit;
User.Text = Configurator.LdapUser;
Catalog.Text = Configurator.LdapDN;
/*Список LDAP-серверов - это массив,
поэтому для обращения к его значениям используется цикл while*/
i: int4 = 0;
while (i <= Configurator.LdapCount)
{
 LDAPHost.Text = Configurator.GetLdapHost(0);
 LDAPPort.Text = String.ToString(Configurator.GetLdapPort(0));
 i += 1;
}
```

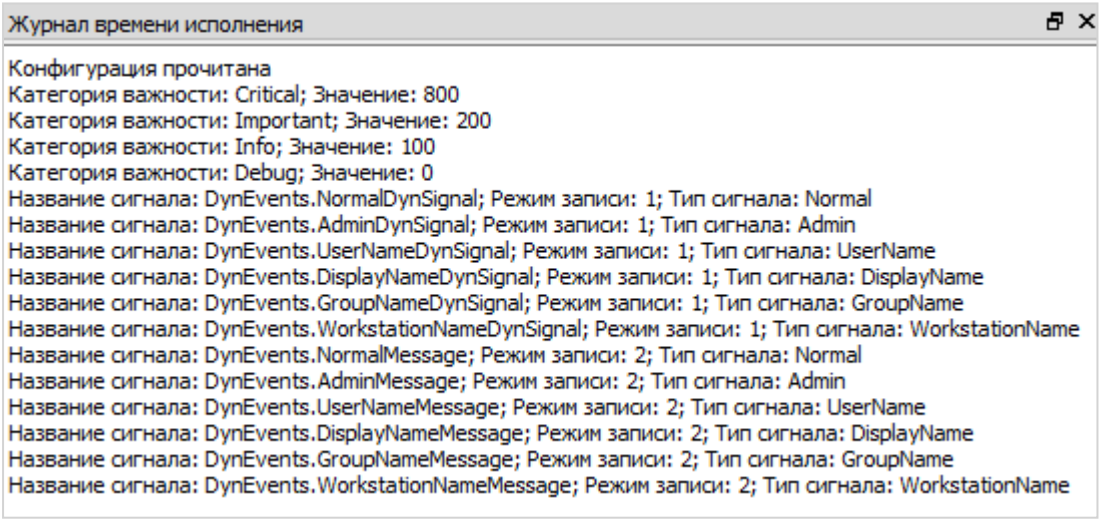
› Команда *WriteAudit* записывает в соответствующие поля информацию о настройках подключения первого описанного в конфигурационном файле сервера-потребителя, а в лог - информацию о категориях важности сообщений и сигналах, предназначенных для записи сообщений в данном сервере. Ознакомьтесь с кодом обработчика события **Invoked** команды:

```
/*здесь извлекается информация только о первом описанном в
конфигурационном файле сервере-потребителе аудита
(индекс во внутреннем массиве элемента Configurator - 0).
Если в вашем конфигурационном файле описано больше серверов-потребителей,
необходимо будет расширить форму и переписать код, добавив вызов
еще одного цикла while*/
AuditHost.Text = Configurator.GetAuditServerHost(0);
AuditPort.Text = String.ToString(Configurator.GetAuditServerPort(0));
AuditProgID.Text = Configurator.GetServerProgId(0);
AuditServerType.Text = Configurator.GetServerType(0);
i: int4 = 0;
while (i < Configurator.GetSeverityCount(0))
{
 DebugTool.Log("Категория важности: "+Configurator.GetSeverityCategory(0,i)+";
Значение: "+String.ToString(Configurator.GetSeverityValue(0,i)));
 i += 1;
}
k: int4 = 0;
while (k < Configurator.GetSignalsCount(0))
{
 DebugTool.Log("Название сигнала: "+Configurator.GetSignalName(0,k)+"; Режим записи:
"+String.ToString(Configurator.GetSignalMode(0,k))+"; Тип сигнала:
"+Configurator.GetSignalType(0,k));
 k += 1;
}
```

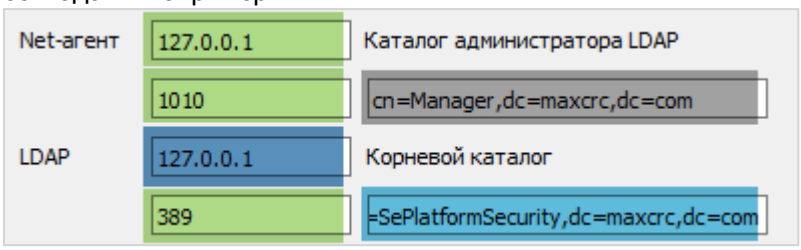
Запустите проект-пример в рантайм. Откроется окно **Конфигурирование агента безопасности**. В поле ввода уже указан путь к конфигурационному файлу службы Агент SePlatform.Security. Нажмите кнопку **Прочитать текущую конфигурацию агента безопасности**. Текстовые поля окна заполнятся информацией о текущих настройках агента безопасности.



В **Журнале времени исполнения** появится информацию о категориях важности сообщений и сигналах, предназначенных для записи сообщений в данном сервере:



Сравните значения в окне проекта со значениями, указанными в конфигурационном файле - они будут совпадать. Например:



seplatform.security.agent.xml

```
1  <SePlatform.Security.Agent>
2      <EntryPointNetAgent Address="127.0.0.1" Port="1010"/>
3      <LdapHosts>
4          <LDAPServer Address="127.0.0.1" Port="389"/>
5      </LdapHosts>
6      <LdapUser value="cn=Manager,dc=maxcrc,dc=com"/>
7      <LdapPassword value="GODP7EWzSfSVC7UmObwbNEzgj9Xh/f+8KYT8"/>
8      <LdapSecure value="False"/>
9      <SecurityDn value="ou=SePlatformSecurity,dc=maxcrc,dc=com"/>
```

Подробнее с остальными свойствами, функциями и событиями компонента **Мастер конфигурирования Security** можно ознакомиться в справочном руководстве (стр. 116).

2. Справочное руководство

2.1. Контекст безопасности (SecurityContext)

Компонент предназначен для взаимодействия с подсистемой безопасности SePlatform.Security. Это позволяет:

- осуществлять вход с учетными данными и выход;
- управлять информацией о текущем пользователе: разрешениями, запретами, учетными данными, пользовательскими сессиями;
- получать информацию о соединении с Агент SePlatform.Security;

Кроме того, компонент обеспечивает обмен данными между остальными компонентами расширения SePlatform.HMI.Security и подсистемой безопасности SePlatform.Security.

2.1.1. Свойства

CurrentUser

Логин текущего пользователя. Тип значения - string.

Только для чтения в режиме рантайма.

ПРИМЕР

Вызов: SecurityContext.CurrentUser

Пример значения: **«ivanov»**.

ПРИМЕЧАНИЕ

С момента ввода корректных учетных данных подсистема безопасности регистрирует вошедшего как текущего пользователя. Для всех свойств, функций и событий предоставляется контекст текущего пользователя.

CurrentUserId

Уникальный идентификатор (uid) текущего пользователя. Тип значения - string.

Только для чтения в режиме рантайма.

ПРИМЕР

Вызов: SecurityContext.CurrentUserId

Вид значения: alpha:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx

Пример значения: **«alpha:c3a997b1-326e-42ee-8907-e359fc17feb7»**.

ПРИМЕЧАНИЕ

Уникальный идентификатор формируется подсистемой безопасности в момент создания учетной записи.

CurrentUserDisplayName

Отображаемое имя текущего пользователя. Тип значения - string.

Только для чтения в режиме рантайма.

ПРИМЕР

Вызов: SecurityContext.CurrentUserDisplayName

Пример значения: **«Иванов»**.

GuestMode

Признак гостевого режима. Тип значения - bool.

Только для чтения в режиме рантайма.

ПРИМЕР

Вызов: SecurityContext.CurrentUser

Значение:

- **«true»** - текущим пользователем является пользователь по умолчанию;
- **«false»** - текущий пользователь вошел с личными учетными данными.

Connected

Состояние соединения с Агент SePlatform.Security. Тип значения - bool.

Только для чтения в режиме рантайма.

ПРИМЕР

Вызов: SecurityContext.Connected

Значение:

- **«true»** - есть соединение с Агент SePlatform.Security;
- **«false»** - нет соединения с Агент SePlatform.Security.

ConnectionError

Текст ошибки соединения с Агент SePlatform.Security. Тип значения - string.

Только для чтения в режиме рантайма.

ПРИМЕР

Вызов: SecurityContext.ConnectionError

Пример значения: **«Соединение с Net агентом разорвано. Код ошибки1»**.

SessionStartTime

Метка времени начала текущей сессии пользователя. Тип значения - timestamp.

Только для чтения в режиме рантайма.

ПРИМЕР

Пример вызова: Text_Msg.Text = DateTime.ToString(SecurityContext.SessionStartTime)

Пример значения: «**13.08.2021 09:20:14**» после применения функции DateTime.ToString().

InactiveRemainTime

Оставшееся время неактивности пользователя. Тип значения - uint4, единица измерения - секунда.

Только для чтения в режиме рантайма.

ПРИМЕЧАНИЕ

В момент, когда пользователь перестает взаимодействовать с АРМ, запускается таймер, отсчитывающий оставшееся время неактивности.

Значение максимального времени неактивности для текущего пользователя указано в праве **Максимальное время бездействия, мин (MaxIdleTime)**. Назначить право можно в Конфигуратор SePlatform.Security.

По истечении указанного в праве времени сессия текущего пользователя автоматически завершается.

ПРИМЕР

Вызов: SecurityContext.InactiveRemainTime

ПРИМЕЧАНИЕ

Для отслеживания активности должен выполняться процесс SePlatform.Security CheckActivity Executable, запускаемый утилитой `seplatform.security.useractivity.exe`.

SessionDurationLimit

Максимальная длительность сессии текущего пользователя. Тип значения - uint4, единица измерения - секунда.

Только для чтения в режиме рантайма.

ПРИМЕЧАНИЕ

Значение максимальной длительности сессии для текущего пользователя указано в праве **Максимальное время сессии, мин (SessionDurationLimit)**. Назначить право можно в Конфигуратор SePlatform.Security.

ПРИМЕР

Вызов: SecurityContext.SessionDurationLimit

Пример значения: **«120»**.

SessionExpiresIn

Оставшееся время до завершения текущей сессии. Тип значения - uint4, единица измерения - секунда.

Только для чтения в режиме рантайма.

ПРИМЕЧАНИЕ

В момент, когда пользователь совершает вход с личными учетными данными, запускается таймер, отсчитывающий оставшееся время до завершения сессии.

Значение максимального времени длительности сессии для текущего пользователя указано в праве **Максимальное время сессии, мин (SessionDurationLimit)**. Назначить право можно в Конфигуратор SePlatform.Security.

По истечении указанного в праве времени сессия текущего пользователя автоматически завершается.

ПРИМЕР

Вызов: SecurityContext.SessionExpiresIn

Пример использования: Уведомление об ограничении длительности сессии (стр. 18).

PasswordExpires

Наличие ограничений срока действия пароля. Тип значения - bool.

Только для чтения в режиме рантайма.

ПРИМЕЧАНИЕ

Значение определяется наличием у текущего пользователя права **Срок действия пароля, дней (PasswordAge)**. Назначить право можно в Конфигуратор SePlatform.Security.

ПРИМЕР

Вызов: SecurityContext.PasswordExpires

Значение:

- **«true»** - есть ограничения срока действия пароля;
- **«false»** - нет ограничений срока действия пароля.

PasswordExpiresSoon

Уведомление о скором истечении срока действия пароля. Тип значения - bool.

Только для чтения в режиме рантайма.

ПРИМЕЧАНИЕ

Значение свойства связано со значениями прав **Срок действия пароля, дней (PasswordAge)** и **Уведомление о смене пароля, дней (PasswordNotifyForChange)** текущего пользователя. Оставшееся время действия пароля сравнивается со сроком уведомления о смене пароля.

Назначить права можно в Конфигуратор SePlatform.Security.

ПРИМЕР

Вызов: SecurityContext.PasswordExpiresSoon

Значение:

- **«true»** - оставшееся время действия пароля внутри срока уведомления о смене пароля;
- **«false»** - оставшееся время действия пароля вне срока уведомления о смене пароля.

PasswordExpiresIn

Оставшееся время действия пароля. Тип значения - uint4, единица измерения - секунда.

Только для чтения в режиме рантайма.

ПРИМЕЧАНИЕ

В момент, когда пользователь меняет пароль, запускается таймер, отсчитывающий оставшееся время действия пароля.

Значение срока действия пароля для текущего пользователя указано в праве **Срок действия пароля, дней (PasswordAge)**. Назначить право можно в Конфигуратор SePlatform.Security.

По истечении указанного в праве времени попытки входа пользователя отклоняются подсистемой безопасности SePlatform.Security. Узнать о том, что срок действия пароля истек, можно с помощью события **LoginRejected** компонента **Контекст безопасности** (стр. 58).

ПРИМЕР

Вызов: SecurityContext.PasswordExpiresIn

Пример использования: Уведомление об истечении срока действия пароля (стр. 20).

LoginError

Текст ошибки входа. Тип значения - string.

Только для чтения в режиме рантайма.

ПРИМЕР

Вызов: SecurityContext.PasswordExpiresIn

Пример значения: **«На узле 'LOCAL' зафиксирована ошибка: Служба 'SECURITYAGENT' не зарегистрирована»**.

LoginRejectReason

Причина отклонения входа подсистемой безопасности. Тип значения - string.

Только для чтения в режиме рантайма.

ПРИМЕЧАНИЕ

Подсистема безопасности отклоняет вход при:

- вводе неверных учетных данных;
- истечении срока действия пароля;
- отсутствии прав на вход.

ПРИМЕР

Вызов: SecurityContext.LoginRejectReason

Пример значения: **«Неверные учетные данные»**.

PasswordChangeError

Текст ошибки, возникающей при попытке смены пароля. Тип значения - string.

Только для чтения в режиме рантайма.

ПРИМЕР

Вызов: SecurityContext.PasswordChangeError

Пример значения: **«Не вышел минимальный срок действия пароля»**.

GroupCount

Количество групп, в которых состоит текущий пользователь. Тип значения - uint8.

Только для чтения в режиме рантайма.

ПРИМЕР

Вызов: SecurityContext.GroupCount

Пример значения: **«2»**.

2.1.2. Функции

Login

```
void Login(string login, string password)
```

Выполняет попытку входа.

Активирует событие LoginStarted (стр. 58).

Входные параметры:

- логин (тип значения - string);
- пароль (тип значения - string).

ПРИМЕР

Вызов: SecurityContext.Login("login", "password")

Результат:

- вход произведен;
- вход не произведен.

ОБРАТИТЕ ВНИМАНИЕ

Исполнение функции останавливает другие процессы в проекте до тех пор, пока вход не будет выполнен или не возникнет ошибка входа. Поэтому проект в режиме исполнения может зависать. Для входа лучше использовать функцию **AsyncLogin**.

AsyncLogin

```
void AsyncLogin(string login, string password)
```

Выполняет попытку асинхронного входа.

Активирует событие LoginStarted (стр. 58).

Входные параметры:

- логин (тип значения - string);
- пароль (тип значения - string).

ПРИМЕР

Вызов: SecurityContext.AsyncLogin("login", "password")

Результат:

- вход произведен;
- вход не произведен.

AsyncLoginWithPasswordChange

```
void AsyncLoginWithPasswordChange(string login, string oldPassword, string newPassword)
```

Выполняет попытку асинхронного входа с изменением пароля.

Активирует событие LoginStarted (стр. 58).

Входные параметры:

- логин (тип значения - string);
- старый пароль (тип значения - string);
- новый пароль (тип значения - string).

ПРИМЕР

Вызов: SecurityContext.AsyncLoginWithPasswordChange("login", "oldpassword", "newpassword")

Результат:

- пароль изменен, вход произведен;
- пароль не изменен, вход не произведен.

ПРИМЕЧАНИЕ

Изменить пароль нельзя, если не истек минимальный срок действия пароля, указанный в праве **Срок действия пароля, дней (PasswordAge)**.

ChangePassword

```
bool ChangePassword(string oldPassword, string newPassword)
```

Выполняет попытку изменения пароля для текущего пользователя.

Входные параметры:

- старый пароль (тип значения - string);
- новый пароль (тип значения - string).

ПРИМЕР

Вызов: SecurityContext.ChangePassword("oldpassword", "newpassword")

Результат:

- пароль изменен;
- пароль не изменен.

ПРИМЕЧАНИЕ

Изменить пароль нельзя, если не истек минимальный срок действия пароля, указанный в праве **Срок действия пароля, дней (PasswordAge)**.

Logout

```
void Logout()
```

Выполняет выход пользователя.

Функция не требует входных параметров.

ПРИМЕР

Вызов: SecurityContext.Logout()

Результат: завершение сессии. Текущим пользователем становится пользователь, чья учетная запись указана по умолчанию.

Group

```
string Group(uint8 groupNumber)
```

Предоставляет название группы, в которой состоит текущий пользователь, по её номеру в списке.

Входной параметр - порядковый номер группы в списке групп текущего пользователя (тип значения - uint8).

ПРИМЕЧАНИЕ

Нумерация групп пользователя в списке начинается с **«0»**.

ПРИМЕР

Вызов: SecurityContext.Group(0)

GroupDisplayName

```
string GroupDisplayName(uint8 groupNumber)
```

Предоставляет отображаемое имя группы, в которой состоит текущий пользователь, по её номеру в списке.

Входной параметр - порядковый номер группы в списке групп текущего пользователя (тип значения - uint8).

ПРИМЕЧАНИЕ

Нумерация групп пользователя в списке начинается с **«0»**.

ПРИМЕР

Вызов: SecurityContext.GroupDisplayName(0)

Пример использования:

LogAudit

```
void LogAudit(string message, uint4 importance)
```

Функция, записывающая сообщение аудита с указанной важностью.

ПРИМЕЧАНИЕ

Аудит - это запись сообщений от подсистемы безопасности в базу данных на сервере.

Входные параметры:

- сообщение, которое необходимо отправить на аудит (тип значения - string);
- уровень важности сообщения, оцененный по четрехбальной шкале, где 0 - наиболее важное сообщение, 3 - наименее важное сообщение (тип значения - uint4).

Подробнее настройка и использование аудита сообщений описаны в документации на SePlatform.Security.

ПРИМЕР

Вызов: SecurityContext.LogAudit("Обновил запись в журнале",2)

LogAuditExt

```
void LogAuditExt(string message, uint4 importance, string type)
```

Функция, записывающая сообщение аудита с указанными важностью и типом.

ПРИМЕЧАНИЕ

Аудит - это запись сообщений от подсистемы безопасности в базу данных на сервере.

Входные параметры:

- сообщение, которое необходимо отправить на аудит (тип значения - string);
- уровень важности сообщения, оцененный по четрехбальной шкале, где 0 - наиболее важное сообщение, 3 - наименее важное сообщение (тип значения - uint4);
- тип сообщения. Здесь следует указать одно из значений, описанных при настройке аудита в конфигурационном файле SePlatform.Security - `seplatform.security.agent.xml`. По умолчанию - **«Normal»** или **«Admin»**.

Подробнее настройка и использование аудита сообщений описаны в документации на SePlatform.Security.

ПРИМЕР

Вызов: SecurityContext.LogAuditExt("Обновил запись в журнале",2,"Normal")

ForceStopUserSession

```
void ForceStopUserSession(string login, string netName)
```

Завершает текущую сессию пользователя.

Завершить сессию можно как на локальном АРМ, так и на удаленном АРМ, входящем в сеть SePlatform.Net.

Входные параметры:

- логин учетной записи пользователя (тип значения - string);
- имя узла в сети SePlatform.Net (тип значения - string). Для завершения сессии на локальном АРМ имя узла в сети SePlatform.Net указывать не нужно.

ПРИМЕР

Вызов: SecurityContext.ForceStopUserSession("login","NetName")

ResetUserFailedLoginCounter

```
void ResetUserFailedLoginCounter(string login, string netName)
```

Сбрасывает счетчик неудачных попыток входа пользователя.

Сбросить счетчик можно как для попыток входа на локальном АРМ, так и на удаленном АРМ, входящем в сеть SePlatform.Net.

Входные параметры:

- логин учетной записи пользователя (тип значения - string);
- имя узла в сети SePlatform.Net (тип значения - string). Для сброса счетчика на локальном АРМ имя узла в сети SePlatform.Net указывать не нужно.

ПРИМЕЧАНИЕ

Для использования счетчика неудачных попыток входа пользователю должно быть назначено право **`Количество неуспешных попыток входа до временной блокировки, шт (MaxAttemptsCount)`**.

ПРИМЕР

Вызов: SecurityContext.ResetUserFailedLoginCounter("login","NetName")

RemoteGetLoggedUsersList

```
void RemoteGetLoggedUsersList(string netName)
```

Запрашивает список текущих пользователей на удаленной рабочей станции в сети SePlatform.Net.

Входной параметр - имя рабочей станции в сети SePlatform.Net (тип значения - string).

ПРИМЕР

Вызов: SecurityContext.RemoteGetLoggedUsersList("NetName")

Результат:

- в случае успешного завершения операции активируется событие RemoteGetLoggedUsersFinished (стр. 60);
- в случае неуспешного завершения операции активируется событие RemoteGetLoggedUsersFailed (стр. 60).

В случае успешной активации события **RemoteGetLoggedUsersFinished** список поместится во внутреннее хранилище компонента. Чтобы воспользоваться данными из списка, используйте функции **GetRemoteLoggedUserCount()** и **GetRemoteLoggedUserByIndex()**.

GetRemoteLoggedUserCount

```
uint4 GetRemoteLoggedUserCount()
```

Возвращает количество текущих пользователей на удаленной рабочей станции. Может использоваться только после вызова функции RemoteGetLoggedUsersList (стр. 57).

Функция не требует входных параметров.

ПРИМЕР

Вызов: SecurityContext.GetRemoteLoggedUserCount()

GetRemoteLoggedUserByIndex

```
string GetRemoteLoggedUserByIndex(uin4 index)
```

Возвращает логин одного из текущих пользователей на удаленной рабочей станции. Может использоваться только после вызова функции RemoteGetLoggedUsersList (стр. 57).

Входной параметр - порядковый номер пользователя в списке, полученном в результате вызова функции RemoteGetLoggedUsersList() (тип значения - uint4).

ПРИМЕЧАНИЕ

Нумерация пользователей в списке начинается с **«0»**.

ПРИМЕР

Вызов: SecurityContext.GetRemoteLoggedUserByIndex(4)

2.1.3. События

CurrentUserChanged

Смена текущего пользователя.

Активируется в момент регистрации подсистемой безопасности нового текущего пользователя.

Возвращает логин текущего пользователя в параметр user, тип значения - string.

ОБРАТИТЕ ВНИМАНИЕ

При запуске проекта SePlatform.HMI в рантайм событие активируется дважды.

В первый раз - при начальной инициализации проекта. Второй раз событие активируется, когда агент передает в проект информацию о текущем пользователе.

ПРИМЕР

Пример использования: Использование проекта с правами пользователя по умолчанию (стр. 11).

ConnectedChanged

Смена состояния соединения с Агент SePlatform.Security.

Активируется в момент разрыва или появления соединения с Агент SePlatform.Security.

Возвращает состояние в параметр connected, тип значения - bool. Параметр может принимать значения:

- **«true»** - есть соединение с Агент SePlatform.Security;
- **«false»** - нет соединения с Агент SePlatform.Security.

PasswordExpiration

Уведомление об истечении срока действия пароля.

Активируется при входе с учетными данными в случае, если оставшееся время действия пароля находится внутри срока уведомления о смене пароля.

Возвращает время, оставшееся до окончания действия пароля, в параметр timeRemain, тип значения - uint4.

ПРИМЕЧАНИЕ

Событие можно использовать при наличии у текущего пользователя права **Уведомление о смене пароля, дней (PasswordNotifyForChange)**. Назначить право можно в Конфигуратор SePlatform.Security.

LoginFailed

Возникновение ошибки при попытке входа.

Активируется при проблемах соединения с Агент SePlatform.Security.

Возвращает текст ошибки в параметр errorMessage, тип значения - string.

LoginStarted

Запуск попытки входа.

Активируется в момент передачи введенных учетных данных подсистеме безопасности SePlatform.Security.

LoginRejected

Отклонение попытки входа подсистемой безопасности SePlatform.Security.

Активируется в момент входа при:

- вводе неверных учетных данных;
- истечении срока действия пароля;
- отсутствии прав на вход.

ПРИМЕР

Пример использования: Демонстрация разграничения доступа в проектах (стр. 9).

AuditFailed

Ошибка аудита сообщений.

ПРИМЕЧАНИЕ

Аудит - это запись сообщений в базу данных на сервере.

Возвращает текст ошибки в параметр errorMessage, тип значения - string.

ForceStopUserSessionFinished

Текущая сессия пользователя завершена.

Активируется в случае успешного завершения операции ForceStopUserSession() (стр. 52).

ResetUserFailedLoginCounterFinished

Счетчик неудачных попыток входа пользователя сброшен.

Активируется в случае успешного завершения операции ResetUserFailedLoginCounter() (стр. 52).

RemoteGetLoggedUsersFinished

Получен список текущих пользователей на удаленной рабочей станции в сети SePlatform.Net.

Активируется в случае успешного завершения операции RemoteGetLoggedUsersList (стр. 57). Список помещается во внутреннее хранилище компонента. Чтобы воспользоваться данными из списка, используйте функции GetRemoteLoggedUserCount (стр. 57) и GetRemoteLoggedUserByIndex (стр. 58).

RemoteGetLoggedUsersFailed

Не удалось получить список текущих пользователей на удаленной рабочей станции в сети SePlatform.Net.

Активируется в случае неуспешного завершения операции RemoteGetLoggedUsersList (стр. 57).

Возвращает номер ошибки в переменную errorCode. Переменная может принимать следующие значения:

- **«1»** - не удалось отправить запрос на удаленную машину.
- **«2»** - имя удаленной машины совпадает с именем локальной.

2.2. Список пользователей (UserList)

Компонент, позволяющий:

- извлекать список пользователей подсистемы безопасности SePlatform.Security;
- обновлять извлеченный список пользователей.

2.2.1. Свойства

Контекст безопасности

Ссылка на элемент типа **Контекст безопасности**, обеспечивающий взаимодействие с подсистемой безопасности SePlatform.Security.

ОБРАТИТЕ ВНИМАНИЕ

Необходимо заполнить для взаимодействия с подсистемой безопасности SePlatform.Security.

Count

Количество пользователей в списке, извлеченном из подсистемы безопасности. Тип значения - uint8.

Только для чтения в режиме рантайма.

ПРИМЕР

Вызов: UserList.Count

Пример использования: Вход с личными учетными данными (стр. 12).

IsUpdatePending

Статус обновления списка пользователей. Тип значения - bool.

Только для чтения в режиме рантайма.

ПРИМЕР

Вызов: UserList.IsUpdatePending

Значение:

- **«true»** - список пользователей обновляется;
- **«false»** - список пользователей не обновляется.

HasError

Наличие ошибок при извлечении или обновлении списка пользователей. Тип значения - bool.

Только для чтения в режиме рантайма.

ПРИМЕР

Вызов: UserList.HasError

Значение:

- **«true»** - есть ошибки;
- **«false»** - нет ошибок.

Error

Текст ошибки, возникшей при извлечении или обновлении списка пользователей. Тип значения - string.

Только для чтения в режиме рантайма.

ПРИМЕР

Вызов: UserList.Error

2.2.2. Функции

GetLoginName

```
string GetLoginName(uint8 number)
```

Предоставляет логин пользователя по номеру в списке (тип значения - string).

Входной параметр - порядковый номер логина пользователя в списке (тип значения - uint8).

ПРИМЕЧАНИЕ

Нумерация логинов пользователей в списке начинается с **«0»**.

ПРИМЕР

Вызов: UserList.GetLoginName(0)

Пример использования - заполнение компонента **ComboBox** логинами всех пользователей подсистемы безопасности в обработчике события UpdateFinished экземпляра **UserList**:

```
i: var = 0;
while (i < Count)
//для всех учетных записей (Count)
{
ComboBox_UserList.AddItem(GetLoginName(i));
//добавление в выпадающий список всех имен пользователей
i+=1;
}
```

GetDisplayName

```
string GetDisplayName(uint8 number)
```

Предоставляет имя пользователя по номеру в списке (тип значения - string).

Входной параметр - порядковый номер имени пользователя в списке (тип значения - uint8).

ПРИМЕЧАНИЕ

Нумерация имён пользователей в списке начинается с **«0»**.

ПРИМЕР

Вызов: UserList.GetDisplayName(0)

Пример использования - заполнение компонента **ComboBox** отображаемыми именами всех пользователей подсистемы безопасности в обработчике события UpdateFinished экземпляра **UserList**:

```
i: var = 0;
while (i < Count)
//для всех учетных записей (Count)
{
ComboBox_UserList.AddItem(GetDisplayName(i));
//добавление в выпадающий список всех имен пользователей
i+=1;
}
```

BeginUpdate

```
void BeginUpdate()
```

Запускает извлечение или обновление списка пользователей.

Функция не требует входных параметров.

ПРИМЕР

Вызов: UserList.BeginUpdate()

Результат: активация события UpdateStarted (стр. 63).

2.2.3. События

UpdateStarted

Запущено извлечение или обновление списка пользователей.

Активируется в результате успешного завершения операции BeginUpdate() (стр. 62).

UpdateFinished

Извлечение или обновление списка пользователей завершено.

ПРИМЕР

Пример использования: Вход с личными учетными данными (стр. 12).

2.3. Настройка безопасности: Контроль целостности (SecurityIntegrityControl)

Подсистема безопасности SePlatform.Security позволяет контролировать целостность указанных файлов и папок. На основе контрольной суммы файлов и папок создается эталон состояния. Когда необходимо проверить их целостность, считается текущая контрольная сумма и сравнивается с эталонным значением.

Компонент **Настройка безопасности: Контроль целостности** предназначен для:

- создания эталона состояния файлов и папок;
- выполнения проверок состояния файлов и папок;
- получения информации об изменениях в файлах и папках.

ПРИМЕР

Сценарий применения компонента описан в разделе 1.6. Контроль целостности (стр. 33).

2.3.1. Свойства

Контекст безопасности

Ссылка на элемент типа **Контекст безопасности**, обеспечивающий взаимодействие с подсистемой безопасности SePlatform.Security.

ОБРАТИТЕ ВНИМАНИЕ

Необходимо заполнить для взаимодействия с подсистемой безопасности SePlatform.Security.

IC_List_JSON

Результат проверки целостности файлов и папок. Предоставляется в формате JSON. Тип значения - string.

Только для чтения в режиме рантайма.

ПРИМЕР

Вызов: SecurityIntegrityControl.IC_List_JSON

Результат:


```
{
"date_last_check": "01.01.2021 00:00:01",
"creator": "Guest(name)",
"date_etalon_created": "01.01.2021 00:00:00",
"data": [
 {
  "ID": "0",
  "ParentID": "-1",
  "Dir": "1",
  "hasRefValue": "0",
  "name": "C:\\Users\\controled objects",
  "MD5_onStart": "",
  "date_onStart": "01.01.2021 00:00:00",
  "status": "4",
  "MD5_current": "",
  "date_current": "",
  "MD5_etalon": "",
  "date_etalon": ""
 },
 {
  "ID": "1",
  "ParentID": "0",
  "Dir": "0",
  "hasRefValue": "0",
  "name": "ex1.txt",
  "MD5_onStart": "cadf2174b6ff3f2d027ee2393ff0a392",
  "date_onStart": "01.01.2021 00:00:00",
  "status": "4",
  "MD5_current": "cadf2174b6ff3f2d027ee2393ff0a392",
  "date_current": "01.01.2021 00:00:00",
  "MD5_etalon": "",
  "date_etalon": ""
 }
]
}
```


Основной является следующая информация:

- date_last_check - дата последней проверки целостности;
- date_etalon_created - дата создания эталона;
- Dir - флаг директории:
  - **«1»**: проверяемый объект - директория;
  - **«0»**: проверяемый объект - файл;
- status - состояние проверяемого объекта:

- **«0»** - ошибок нет;
- **«4»** - файл не существует, либо указана символическая ссылка на несуществующий файл/директорию, либо недостаточно прав для чтения файла;
- **«6»** - изменилась дата изменения файла, при этом контрольная сумма файла совпадает с эталонной;
- **«8»** - нарушена целостность файла (контрольные суммы не совпадают);
- **«16»** - обнаружен новый файл (для него нет эталонного значения).

2.3.2. Функции

Update_IC

```
void Update_IC()
```

Выполняет проверку целостности файлов и папок:

- вычисляет текущую контрольную сумму;
- сравнивает результат с эталонным значением.

Функция не требует входных параметров.

ПРИМЕР

Вызов: SecurityIntegrityControl.Update_IC()

Результат:

- в случае успешного завершения операции активируется событие UpdateFinished (стр. 68);
- в случае неуспешного завершения операции активируется событие UpdateFailed (стр. 68).

Get_IC_List

```
void Get_IC_List()
```

Запрашивает результат проверки целостности файлов и папок.

Функция не требует входных параметров.

ПРИМЕР

Вызов: SecurityIntegrityControl.Get_IC_List()

Результат:

- в случае успешного завершения операции активируется событие ListIsReady (стр. 68);
- в случае неуспешного завершения операции активируется событие GetListFailed (стр. 68).

Create_IC_Etalon

```
void Create_IC_Etalon()
```

Создает эталон на основе текущего состояния файлов и папок:

- вычисляет текущую контрольную сумму;
- сохраняет значение в качестве эталонного.

Функция не требует входных параметров.

ПРИМЕР

Вызов: SecurityIntegrityControl.Create_IC_Etalon()

Результат:

- в случае успешного завершения операции активируется событие CreateFinished (стр. 68);
- в случае неуспешного завершения операции активируется событие CreateFailed (стр. 68).

UpdateRemote_IC

```
void UpdateRemote_IC(string netName)
```

Аналогично функции Update_IC(), но на удаленной рабочей станции в сети SePlatform.Net.

Входной параметр - имя рабочей станции в сети SePlatform.Net (тип значения - string).

ПРИМЕР

Вызов: SecurityIntegrityControl.UpdateRemote_IC("Netname")

Результат:

- в случае успешного завершения операции активируется событие RemoteUpdateFinished (стр. 68);
- в случае неуспешного завершения операции активируется событие RemoteUpdateFailed (стр. 68).

GetRemote_IC_List

```
void GetRemote_IC_List(string netName)
```

Аналогично функции Get_IC_List(), но на удаленной рабочей станции в сети SePlatform.Net.

Входной параметр - имя рабочей станции в сети SePlatform.Net (тип значения - string).

ПРИМЕР

Вызов: SecurityIntegrityControl.GetRemote_IC_List("Netname")

Результат:

- в случае успешного завершения операции активируется событие RemoteListIsReady (стр. 68);
- в случае неуспешного завершения операции активируется событие RemoteGetListFailed (стр. 68).

CreateRemote_IC_Etalon

```
void CreateRemote_IC_Etalon(string netName)
```

Аналогично функции Create_IC_Etalon(), но на удаленной рабочей станции в сети SePlatform.Net.

Входной параметр - имя рабочей станции в сети SePlatform.Net (тип значения - string).

ПРИМЕР

Вызов: SecurityIntegrityControl.CreateRemote_IC_Etalon("Netname")

Результат:

- в случае успешного завершения операции активируется событие RemoteCreateFinished (стр. 68);
- в случае неуспешного завершения операции активируется событие RemoteCreateFailed (стр. 68).

2.3.3. События

UpdateFinished

Выполнена проверка целостности файлов.

Активируется в случае успешного завершения операции Update_IC() (стр. 66).

UpdateFailed

Не удалось выполнить проверку целостности файлов.

Активируется в случае неуспешного завершения операции Update_IC() (стр. 66).

Возвращает текст ошибки в переменную errorMessage.

CreateFinished

Создан эталон состояния файлов.

Активируется в случае успешного завершения операции Create_IC_Etalon() (стр. 66).

CreateFailed

Не удалось создать эталон состояния файлов.

Активируется в случае неуспешного завершения операции Create_IC_Etalon() (стр. 66).

Возвращает текст ошибки в переменную errorMessage.

ListIsReady

Получен результат проверки целостности файлов.

Активируется в случае успешного завершения операции Get_IC_List() (стр. 66).

Возвращает данные в JSON-формате в переменную IC_List_JSON.

ПРИМЕР

Результат проверки в JSON-формате:


```
{
"date_last_check": "01.01.2021 00:00:01",
"creator": "Guest(name)",
"date_etalon_created": "01.01.2021 00:00:00",
"data": [
 {
  "ID": "0",
  "ParentID": "-1",
  "Dir": "1",
  "hasRefValue": "0",
  "name": "C:\\Users\\controled objects",
  "MD5_onStart": "",
  "date_onStart": "01.01.2021 00:00:00",
  "status": "4",
  "MD5_current": "",
  "date_current": "",
  "MD5_etalon": "",
  "date_etalon": ""
 },
 {
  "ID": "1",
  "ParentID": "0",
  "Dir": "0",
  "hasRefValue": "0",
  "name": "ex1.txt",
  "MD5_onStart": "cadf2174b6ff3f2d027ee2393ff0a392",
  "date_onStart": "01.01.2021 00:00:00",
  "status": "4",
  "MD5_current": "cadf2174b6ff3f2d027ee2393ff0a392",
  "date_current": "01.01.2021 00:00:00",
  "MD5_etalon": "",
  "date_etalon": ""
 }
]
}
```


Основной является следующая информация:

- date_last_check - дата последней проверки целостности;
- date_etalon_created - дата создания эталона;
- Dir - флаг директории:
  - **«1»**: проверяемый объект - директория;
  - **«0»**: проверяемый объект - файл;
- status - состояние проверяемого объекта:
  - **«0»** - ошибок нет;
  - **«4»** - файл не существует;
  - **«8»** - нарушена целостность файла (контрольные суммы не совпадают).

GetListFailed

Не удалось получить результат проверки целостности файлов.

Активируется в случае неуспешного завершения операции Get_IC_List() (стр. 66).

Возвращает текст ошибки в переменную errorMessage.

RemoteUpdateFinished

Выполнена проверка целостности файлов на удаленной рабочей станции в сети SePlatform.Net.

Активируется в случае успешного завершения операции UpdateRemote_IC() (стр. 66).

RemoteUpdateFailed

Не удалось выполнить проверку целостности файлов на удаленной рабочей станции в сети SePlatform.Net.

Активируется в случае неуспешного завершения операции UpdateRemote_IC() (стр. 66).

Возвращает текст ошибки в переменную errorMessage

RemoteCreateFinished

Создан эталон состояния файлов на удаленной рабочей станции в сети SePlatform.Net.

Активируется в случае успешного завершения операции CreateRemote_IC_Etalon (стр. 66).

RemoteCreateFailed

Не удалось создать эталон состояния файлов на удаленной рабочей станции в сети SePlatform.Net.

Активируется в случае неуспешного завершения операции CreateRemote_IC_Etalon (стр. 66).

Возвращает текст ошибки в переменную errorMessage.

RemoteListIsReady

Получен результат проверки целостности файлов на удаленной рабочей станции в сети SePlatform.Net.

Активируется в случае успешного завершения операции GetRemote_IC_List (стр. 66).

Возвращает данные в JSON-формате в переменную IC_List_JSON.

RemoteGetListFailed

Не удалось получить результат проверки целостности файлов на удаленной рабочей станции в сети SePlatform.Net.

Активируется в случае неуспешного завершения операции GetRemote_IC_List (стр. 66).

Возвращает текст ошибки в переменную errorMessage.

2.4. Строковый элемент безопасности (StringTokenProxy)

Компонент, обеспечивающий связь со строковым правом подсистемы безопасности SePlatform.Security.

Связь позволяет получать разрешения и запреты для текущего пользователя.

2.4.1. Свойства

Контекст безопасности

Ссылка на элемент типа **Контекст безопасности**, обеспечивающий взаимодействие с подсистемой безопасности SePlatform.Security.

ОБРАТИТЕ ВНИМАНИЕ

Необходимо заполнить для взаимодействия с подсистемой безопасности SePlatform.Security.

Приложение (Application)

Имя приложения, к которому относится право, связанное с элементом. Тип значения - string.

ПРИМЕР

Вызов: StringTokenProxy.Application

Пример значения: **«Управление состоянием оборудования»**.

Право (Right)

Имя права, с которым связан элемент. Тип значения - string.

ПРИМЕР

Вызов: StringTokenProxy.Right

Пример значения: **«Управление резервуаром»**.

AllowedCount

Количество разрешений текущего пользователя в праве, связанном с элементом. Тип значения - uint4.

Только для чтения в режиме рантайма.

ПРИМЕР

Вызов: StringTokenProxy.AllowedCount

Пример значения: **«1»**.

ForbiddenCount

Количество запретов текущего пользователя в праве, связанном с элементом. Тип значения - uint4.

Только для чтения в режиме рантайма.

ПРИМЕР

Вызов: StringTokenProxy.ForbiddenCount

Пример значения: **«2»**.

Connected

Состояние подписки на право. Тип значения - bool.

Подписка - это состояние соединения элемента с указанным правом.

Только для чтения в режиме рантайма.

ПРИМЕР

Вызов: StringTokenProxy.Connected

Значение:

- **«true»** - подписка активна;
- **«false»** - подписка неактивна.

Error

Ошибка подписки. Тип значения - string.

Только для чтения в режиме рантайма.

ПРИМЕР

Вызов: StringTokenProxy.Error

Пример значения: **«Не удалось получить значение токена права. Объект не найден в хранилище LDAP»**

2.4.2. Функции

GetAllowed

```
string GetAllowed(uint8 number)
```

Предоставляет разрешение текущего пользователя из права, связанного с элементом.

Входной параметр - порядковый номер разрешения в списке разрешений текущего пользователя (тип значения - uint8).

ПРИМЕЧАНИЕ

Нумерация разрешений начинается с **«0»**.

GetForbidden

```
string GetForbidden(uint8 number)
```

Предоставляет запрет текущего пользователя из права, связанного с элементом.

Входной параметр - порядковый номер запрета в списке запретов текущего пользователя (тип значения - uint8).

ПРИМЕЧАНИЕ

Нумерация запретов начинается с **«0»**.

2.4.3. События

ValueChanged

Смена текущего значения права.

Возвращает текущее значение в параметр value, тип значения - bool.

ConnectedChanged

Смена состояния подписки на право.

Возвращает текущее значение в параметр connected, тип значения - bool.

2.5. Булевский элемент безопасности (BoolTokenProxy)

Компонент, обеспечивающий связь с логическим правом подсистемы безопасности SePlatform.Security.

Связь позволяет отслеживать изменение значения права для текущего пользователя.

2.5.1. Свойства

Контекст безопасности

Ссылка на элемент типа **Контекст безопасности**, обеспечивающий взаимодействие с подсистемой безопасности SePlatform.Security.

ОБРАТИТЕ ВНИМАНИЕ

Необходимо заполнить для взаимодействия с подсистемой безопасности SePlatform.Security.

Приложение (Application)

Имя приложения, к которому относится право, связанное с элементом. Тип значения - string.

ПРИМЕР

Вызов: BoolTokenProxy.Application

Пример значения: **«Управление состоянием оборудования»**.

Право (Right)

Имя права, с которым связан элемент. Тип значения - string.

ПРИМЕР

Вызов: BoolTokenProxy.Right

Пример значения: **«Управление насосом»**.

Value

Значение указанного права для текущего пользователя. Тип значения - bool.

Только для чтения в режиме рантайма.

ПРИМЕР

Вызов: BoolTokenProxy.Value

Значение:

- **«true»** - доступно текущему пользователю;
- **«false»** - недоступно текущему пользователю.

Connected

Состояние подписки на право. Тип значения - bool.

Подписка - это состояние соединения элемента с указанным правом.

Только для чтения в режиме рантайма.

ПРИМЕР

Вызов: BoolTokenProxy.Connected

Значение:

- **«true»** - подписка активна;
- **«false»** - подписка неактивна.

Error

Ошибка подписки. Тип значения - string.

Только для чтения в режиме рантайма.

ПРИМЕР

Вызов: BoolTokenProxy.Error

Пример значения: **«Не удалось получить значение токена права. Объект не найден в хранилище LDAP»**

2.5.2. События

ValueChanged

Смена текущего значения права.

Возвращает текущее значение в параметр value, тип значения - bool.

ConnectedChanged

Смена состояния подписки на право.

Возвращает текущее значение в параметр connected, тип значения - bool.

2.6. Настройка безопасности: Менеджер (SecurityManager)

Компонент предназначен для:

- получения списков учетных записей, групп пользователей и приложений;
- удаления учетных записей, групп пользователей и приложений;
- создания и восстановления резервных копий конфигурации SePlatform.Security.

Используется для конфигурирования подсистемы безопасности SePlatform.Security из проектов SePlatform.HMI.

2.6.1. Свойства

Контекст безопасности

Ссылка на элемент типа **Контекст безопасности**, обеспечивающий взаимодействие с подсистемой безопасности SePlatform.Security.

ОБРАТИТЕ ВНИМАНИЕ

Необходимо заполнить для взаимодействия с подсистемой безопасности SePlatform.Security.

AgentStatus

Текущее состояние Агент SePlatform.Security. Тип значения - uint1.

Только для чтения в режиме рантайма.

ПРИМЕР

Вызов: SecurityManager.AgentStatus.

2.6.2. Функции

RequestAppList

```
void RequestAppList()
```

Запрашивает список всех приложений, имеющихся в подсистеме безопасности SePlatform.Security.

Функция не требует входных параметров.

ОБРАТИТЕ ВНИМАНИЕ

Просмотр списка приложений доступен только пользователям с правами администратора.

ПРИМЕР

Вызов: SecurityManager.RequestAppList()

Результат:

- в случае успешного завершения операции активируется событие RequestAppListComplete (стр. 81);
- в случае неуспешного завершения операции активируется событие RequestAppListFailed (стр. 81).

RequestGroupList

```
void RequestGroupList()
```

Запрашивает список всех групп пользователей, имеющихся в подсистеме безопасности SePlatform.Security.

Функция не требует входных параметров.

ОБРАТИТЕ ВНИМАНИЕ

Просмотр списка групп пользователей доступен только пользователям с правами администратора.

ПРИМЕР

Вызов: SecurityManager.RequestGroupList()

Результат:

- в случае успешного завершения операции активируется событие RequestGroupListComplete (стр. 81);
- в случае неуспешного завершения операции активируется событие RequestGroupListFailed (стр. 81).

RequestUsersList

```
void RequestUsersList()
```

Запрашивает список всех учетных записей, имеющихся в подсистеме безопасности SePlatform.Security.

Функция не требует входных параметров.

ОБРАТИТЕ ВНИМАНИЕ

Просмотр списка учетных записей доступен только пользователям с правами администратора.

ПРИМЕР

Вызов: SecurityManager.RequestUsersList()

Результат:

- в случае успешного завершения операции активируется событие RequestUsersListComplete (стр. 81);
- в случае неуспешного завершения операции активируется событие RequestUsersListFailed (стр. 81).

ПРИМЕР

Применение функции подробнее описано в 1.5. Собственный конфигуратор подсистемы безопасности (стр. 22).

DeleteApplication

```
void DeleteApplication(string appName)
```

Удаляет приложение из подсистемы безопасности.

Входной параметр - имя приложения (тип значения - string).

ОБРАТИТЕ ВНИМАНИЕ

Удаление приложений доступно только пользователям с правами администратора.

ПРИМЕР

Вызов: SecurityManager.DeleteApplication("Управление состоянием оборудования")

Результат:

- в случае успешного завершения операции активируется событие DeleteApplicationComplete (стр. 81);
- в случае неуспешного завершения операции активируется событие DeleteApplicationFailed (стр. 81).

DeleteGroup

```
void DeleteGroup(string groupUID)
```

Удаляет группу пользователей из подсистемы безопасности.

Входной параметр - уникальный идентификатор (uid) группы вида alpha:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx (тип значения - string).

ОБРАТИТЕ ВНИМАНИЕ

Удаление групп пользователей доступно только пользователям с правами администратора.

ПРИМЕР

Вызов: SecurityManager.DeleteGroup("alpha:d01969e0-ae05-4548-9375-2533b326a588")

Результат:

- в случае успешного завершения операции активируется событие DeleteGroupComplete (стр. 81);
- в случае неуспешного завершения операции активируется событие DeleteGroupFailed (стр. 81).

DeleteUser

```
void DeleteUser(string userUID)
```

Удаляет четную запись из подсистемы безопасности.

Входной параметр - уникальный идентификатор (uid) учетной записи вида alpha:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx (тип значения - string).

ОБРАТИТЕ ВНИМАНИЕ

Удаление учетных записей доступно только пользователям с правами администратора.

ПРИМЕР

Вызов: SecurityManager.DeleteUser("alpha:d01969e0-ae05-4548-9375-2533b326a588")

Результат:

- в случае успешного завершения операции активируется событие DeleteUserComplete (стр. 81);
- в случае неуспешного завершения операции активируется событие DeleteUserFailed (стр. 81).

GetErrorDescriptionByCode

```
string GetErrorDescriptionByCode(uint1 FailReason)
```

Возвращает текстовое описание ошибок, возникающих при запросе и удалении приложений, групп пользователей и учетных записей.

Входной параметр - переменная FailReason (тип значения - uint1).

ExportConfiguration

```
void ExportConfiguration(string fullPath)
```

Создает (или перезаписывает) файл резервной копии текущей конфигурации SePlatform.Security в указанном месте.

Входной параметр - полный путь к создаваемому файлу вместе с именем файла (тип значения - string).

ПРИМЕР

Вызов: SecurityManager.ExportConfiguration("D:/export.ldb")

Результат:

- в случае успешного завершения операции активируется событие GetConfigurationFinished (стр. 85);
- в случае неуспешного завершения операции активируется событие GetConfigurationFailed (стр. 86).

ImportConfiguration

```
void ImportConfiguration(string fullPath)
```

Восстанавливает резервную копию конфигурации SePlatform.Security из указанного файла.

Входной параметр - полный путь к файлу резервной копии вместе с именем файла (тип значения - string).

ПРИМЕР

Вызов: SecurityManager.ImportConfiguration("D:/export.ldb")

Результат:

- в случае успешного завершения операции активируется событие SetConfigurationFinished (стр. 86);
- в случае неуспешного завершения операции активируется событие SetConfigurationFailed (стр. 86).

2.6.3. События

RequestAppListComplete

Получен список приложений, имеющихся в подсистеме безопасности SePlatform.Security.

Активируется в случае успешного завершения операции RequestAppList().

Возвращает данные в JSON-формате в переменную JSONAppList.

ПРИМЕР

Результат запроса списка приложений в JSON-формате:

```
{
 "data":[
  {
  "applicationName": "Управление состоянием оборудования",
  "applicationType": "CustomApp"
  },
  {
   "applicationName": "SePlatform.Security",
   "applicationType": "SystemApp"
  }
 ]
}
```

В данном примере:

- приложение **Управление состоянием оборудования** создано пользователем с правами администратора, о чем говорит тип приложения CustomApp;
- приложение **SePlatform.Security** - системное приложение, в котором определены системные права.

RequestGroupListComplete

Получен список групп пользователей, имеющихся в подсистеме безопасности SePlatform.Security.

Активируется в случае успешного завершения операции RequestGroupList() (стр. 77).

Возвращает данные в JSON-формате в переменную JSONGroupList.

ПРИМЕР

Результат запроса списка групп в JSON-формате:

```
{
 "data":[
  {
  "description": " ",
  "displayName": "Диспетчеры",
  "uid": "alpha:d01969e0-ae05-4548-9375-2533b326a588"
  },
  {
   "description": " ",
   "displayName": "Операторы",
   "uid": "alpha:de893741-9258-45b2-a61a-e79fd87ae10c"
  }
 ]
}
```

Уникальный идентификатор (uid) формируется подсистемой безопасности в момент создания приложения.

RequestUsersListComplete

Получен список пользователей, имеющихся в подсистеме безопасности SePlatform.Security.

Активируется в случае успешного завершения операции RequestUsersList() (стр. 77).

Возвращает данные в JSON-формате в переменную JSONUsersList.

ПРИМЕР

Результат запроса списка пользователей в JSON-формате:


```
{
 "data": [
  {
   "uid": "alpha:1161a849-8849-43de-9ba0-e57c7045acbf",
   "employeeType": "",
   "departmentNumber": "123",
   "mail": "ivanov@qw.qw",
   "telephoneNumber": "123456",
   "description": "",
   "givenName": "Петр",
   "initials": "Петрович",
   "sn": "Петров",
   "displayName": "Петров Петр Петрович",
   "symbolicId": "petrov",
   "groups": [
    "Операторы"
   ],
   "roles": ""
  },
  {
   "uid": "alpha:8d330867-fe85-4b07-89fa-2a995736ac42",
   "employeeType": "",
   "departmentNumber": "123",
   "mail": "petrov@qw.qw",
   "telephoneNumber": "456789",
   "description": "",
   "givenName": "Иван",
   "initials": "Иванович",
   "sn": "Иванов",
   "displayName": "Иванов Иван",
   "symbolicId": "ivanov",
   "groups": [
    "Диспетчеры"
   ],
   "roles": ""
  }
 ]
}
```


Уникальный идентификатор (uid) формируется подсистемой безопасности в момент создания учетной записи.

ПРИМЕР

Применение события подробнее описано в 1.5. Собственный конфигуратор подсистемы безопасности (стр. 22).

RequestAppListFailed

Ошибка при получении списка приложений подсистемы безопасности SePlatform.Security.

Активируется в случае неуспешного завершения операции RequestAppList().

Возвращает номер ошибки в переменную FailReason, тип значения - uint1.

ПРИМЕЧАНИЕ

Получить текст ошибки можно, вызвав функцию GetErrorDescriptionByCode() (стр. 77).

RequestGroupListFailed

Ошибка при получении списка групп пользователей подсистемы безопасности SePlatform.Security.

Активируется в случае неуспешного завершения операции RequestGroupList() (стр. 77).

Возвращает номер ошибки в переменную FailReason, тип значения - uint1.

ПРИМЕЧАНИЕ

Получить текст ошибки можно, вызвав функцию GetErrorDescriptionByCode() (стр. 77).

RequestUsersListFailed

Ошибка при получении списка пользователей подсистемы безопасности SePlatform.Security.

Активируется в случае неуспешного завершения операции RequestUsersList() (стр. 77).

Возвращает номер ошибки в переменную FailReason, тип значения - uint1.

ПРИМЕЧАНИЕ

Получить текст ошибки можно, вызвав функцию GetErrorDescriptionByCode() (стр. 77).

DeleteApplicationComplete

Приложение удалено из подсистемы безопасности SePlatform.Security.

Активируется в случае успешного завершения операции DeleteApplication() (стр. 77).

DeleteGroupComplete

Группа пользователей удалена из подсистемы безопасности SePlatform.Security.

Активируется в случае успешного завершения операции DeleteGroup() (стр. 77).

DeleteUserComplete

Учетная запись удалена из подсистемы безопасности SePlatform.Security.

Активируется в случае успешного завершения операции DeleteUser() (стр. 77).

DeleteApplicationFailed

Ошибка при удалении приложения из подсистемы безопасности SePlatform.Security.

Активируется в случае неуспешного завершения операции DeleteApplication() (стр. 77).

Возвращает номер ошибки в переменную FailReason, тип значения - uint1.

ПРИМЕЧАНИЕ

Получить текст ошибки можно, вызвав функцию GetErrorDescriptionByCode() (стр. 77).

DeleteGroupFailed

Ошибка при удалении группы пользователей из подсистемы безопасности SePlatform.Security.

Активируется в случае неуспешного завершения операции DeleteGroup() (стр. 77).

Возвращает номер ошибки в переменную FailReason, тип значения - uint1.

ПРИМЕЧАНИЕ

Получить текст ошибки можно, вызвав функцию GetErrorDescriptionByCode() (стр. 77).

DeleteUserFailed

Ошибка при удалении учетной записи из подсистемы безопасности SePlatform.Security.

Активируется в случае неуспешного завершения операции DeleteUser() (стр. 77).

Возвращает номер ошибки в переменную FailReason, тип значения - uint1.

ПРИМЕЧАНИЕ

Получить текст ошибки можно, вызвав функцию GetErrorDescriptionByCode() (стр. 77).

AgentStatusChanged

Изменение текущего состояния Агент SePlatform.Security.

ПРИМЕЧАНИЕ

Все возможные состояния Агент SePlatform.Security описаны в свойстве AgentStatus (стр. 76).

GetConfigurationFinished

Файл резервной копии конфигурации SePlatform.Security создан (или перезаписан) успешно.

Активируется в случае успешного завершения операции ExportConfiguration (стр. 80).

GetConfigurationFailed

Ошибка при создании (или перезаписи) файла резервной копии конфигурации SePlatform.Security.

Активируется в случае неуспешного завершения операции ExportConfiguration (стр. 80).

Возвращает код ошибки в переменную FailReason, тип значения – uint1. Если код ошибки равен:

- **«2»** – не удалось экспортировать конфигурацию из LDAP (например, не существует корневого каталога и т.д.);
- **«3»** – нет права на чтение конфигурации;
- **«4»** – не удалось открыть файл для записи конфигурации.

SetConfigurationFinished

Резервная копия конфигурации SePlatform.Security восстановлена успешно.

Активируется в случае успешного завершения операции ImportConfiguration (стр. 80).

SetConfigurationFailed

Не удалось восстановить резервную копию конфигурации SePlatform.Security.

Активируется в случае неуспешного завершения операции ImportConfiguration (стр. 80).

Возвращает код ошибки в переменную FailReason, тип значения – uint1. Если код ошибки равен:

- **«1»** – файл резервной копии поврежден;
- **«2»** – не удалось зачистить базу данных LDAP перед применением конфигурации;
- **«3»** – произошла внутренняя ошибка LDAP;
- **«4»** – произошла непредвиденная ошибка на стороне агента;
- **«5»** – у пользователя нет права на редактирование конфигурации;
- **«6»** – другие ошибки;
- **«7»** – не удается открыть файл с конфигурацией.

LastActionError

Ошибка выполнения последнего действия.

Активируется в результате внесения в конфигурацию подсистемы безопасности таких изменений, которые могут привести к нарушению работы SePlatform.Security. Таким действием является, например, удаление учетной записи единственного администратора.

Возвращает текст ошибки в переменную ErrorMessage, тип значения – string.

2.7. Настройка безопасности: Приложение (SecurityManagerApplication)

Компонент предназначен для загрузки информации о приложении:

- имеющемся в подсистеме безопасности - для просмотра и изменения этой информации;
- созданном из проекта SePlatform.HMI - для дальнейшей отправки в подсистему безопасности.

2.7.1. Свойства

Менеджер конфигурирования безопасности

Ссылка на элемент типа **Настройка безопасности: Менеджер**, обеспечивающий работу с учетными записями, группами пользователей и приложениями.

ОБРАТИТЕ ВНИМАНИЕ

Содержит свойство Контекст безопасности для указания ссылки на элемент типа **Контекст безопасности**.

Необходимо заполнить для взаимодействия с подсистемой безопасности SePlatform.Security.

Имя приложения (ApplicationName)

Имя приложения, загруженного в элемент. Тип значения - string.

ПРИМЕР

Вызов: SecurityManagerApplication.ApplicationName.

Пример значения: **«Управление состоянием оборудования»**.

ApplicationID

Идентификатор (id) приложения, загруженного в элемент. Тип значения - string.

Совпадает с именем приложения.

Только для чтения в режиме рантайма.

ПРИМЕР

Вызов: SecurityManagerApplication.ApplicationID.

Пример значения: **«Управление состоянием оборудования»**.

IsChanged

Уведомление об изменении в приложении, загруженном в элемент. Тип значения - bool.

Только для чтения в режиме рантайма.

ПРИМЕР

Вызов: SecurityManagerApplication.IsChanged.

Значение:

- **«true»** - приложение было изменено;
- **«false»** - нет изменений в приложении.

ПРИМЕЧАНИЕ

Принимает значение **«false»** при:

- загрузке приложения в элемент;
- сохранении приложения в подсистему безопасности.

2.7.2. Функции

Load

```
void Load(string appID)
```

Загружает информацию о приложении из подсистемы безопасности в элемент.

Входной параметр - ID приложения (тип значения - string).

ПРИМЕР

Вызов: SecurityManagerApplication.Load("Управление состоянием оборудования")

Результат:

- в случае успешного завершения операции активируется событие LoadComplete (стр. 96);
- в случае неуспешного завершения операции активируется событие LoadFailed (стр. 96).

Save

```
void Save()
```

Отправляет информацию о приложении в подсистему безопасности после:

- редактирования ранее выгруженного из подсистемы безопасности приложения;
- создания приложения из проекта SePlatform.HMI с помощью функции New() и свойств элемента.

Функция не требует входных параметров.

ПРИМЕР

Вызов: SecurityManagerApplication.Save()

Результат:

- в случае успешного завершения операции активируется событие SaveComplete (стр. 96);
- в случае неуспешного завершения операции активируется событие SaveFailed (стр. 96).

GetTokensList

```
string GetTokensList()
```

Предоставляет список прав из приложения, загруженного в элемент.

Функция не требует входных параметров.

Возвращает данные в JSON-формате.

ПРИМЕР

Вызов: SecurityManagerApplication.GetTokensList()

Результат запроса списка прав в JSON-формате:


```
{
 "data": [
  {
   "tokenID": "Управление насосом",
   "tokenType": "bool",
   "tokenName": "Управление насосом",
   "tokenDescription": "Право доступа к насосу",
   "tokenValues": [
    {
     "allow": "FALSE",
     "deny": "TRUE",
     "subjects": [
      {
       "type": "group",
       "name": "Диспетчеры"
      }
     ]
    },
    {
     "allow": "TRUE",
     "deny": "FALSE",
     "subjects": [
      {
       "type": "user",
       "name": "petrov"
      },
      {
       "type": "group",
       "name": "Операторы"
      },
      {
       "type": "role",
       "name": "Начальник участка"
      }
     ]
    }
   ]
  }
 ]
}
```


GetRolesList

```
string GetRolesList()
```

Предоставляет список ролей из приложения, загруженного в элемент.

Функция не требует входных параметров.

Возвращает данные в JSON-формате.

ПРИМЕР

Вызов: SecurityManagerApplication.GetRolesList()

Результат запроса списка ролей в JSON-формате:

```
{
 "data": [
  {
   "roleID": "alpha:4ce9483e-e137-41c2-bf1c-343e9b731f20",
   "roleName": "dispatcher"
  }
 ]
}
```

GetRoleRights

```
string GetRoleRights(string roleUID)
```

Предоставляет список прав, назначенных текущей роли.

Входной параметр – уникальный идентификатор (uid) роли вида alpha:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx (тип значения – string).

Возвращает данные в JSON-формате.

ПРИМЕР

Вызов: SecurityManagerApplication.GetRoleRights("alpha:4ce9483e-e137-41c2-bf1c-343e9b731f20")

Результат запроса списка ролей в JSON-формате:


```
{
 "data": [
  {
   "appName": "Управление состоянием оборудования",
   "tokenType": "bool",
   "tokenName": "Управление резервуаром",
   "tokenDescription": "Право управления резервуаром",
   "ownerId": "alpha:4ce9483e-e137-41c2-bf1c-343e9b731f20",
   "tokenValueAllow": "TRUE",
   "tokenValueDeny": "FALSE"
  },
  {
   "appName": "Управление состоянием оборудования",
   "tokenType": "bool",
   "tokenName": "Управление насосом",
   "tokenDescription": "Право управления насосом",
   "ownerId": "alpha:4ce9483e-e137-41c2-bf1c-343e9b731f20",
   "tokenValueAllow": "FALSE",
   "tokenValueDeny": "TRUE"
  }
 ]
}
```


New

```
void New()
```

Освобождает элемент от загруженной в него информации для создания нового приложения.

Функция не требует входных параметров.

ПРИМЕР

Вызов: SecurityManagerApplication.New()

ПРИМЕЧАНИЕ

После создания нового приложения не забудьте загрузить его в подсистему безопасности с помощью функции Save().

CreateToken

```
void CreateToken(uint1 tokenType, string tokenName, string decsription)
```

Создает право в приложении, загруженном в элемент.

Входные параметры:

- тип права (тип значения - uint1):
  - **«0»** - строковое;
  - **«1»** - логическое;
- имя права (тип значения - string);
- описание (тип значения - string).

ПРИМЕР

Вызов: SecurityManagerApplication.CreateToken(1,"right","description")

ChangeToken

```
void ChangeToken(string tokenID, uint1 tokenType, string tokenName, string decsription)
```

Меняет тип, имя и описание права в приложении, загруженном в элемент.

Входные параметры:

- ID права (тип значения - string);
- тип права (тип значения - uint1):
  - **«0»** - строковое;
  - **«1»** - логическое;
- новое имя права (тип значения - string);
- новое описание права (тип значения - string).

ПРИМЕР

Вызов: SecurityManagerApplication.ChangeToken("tokenID",1,"newname","newdescription")

ОБРАТИТЕ ВНИМАНИЕ

Входной параметр тип права не будет требоваться в будущих версиях.

DeleteToken

```
void DeleteToken(string tokenID)
```

Удаляет право из приложения, загруженного в элемент.

Входной параметр - ID права (тип значения - string).

ПРИМЕР

Вызов: SecurityManagerApplication.DeleteToken("right")

CreateRole

```
void CreateRole(string roleName, string appName)
```

Создает роль в указанном приложении.

Входные параметры:

- имя роли (тип значения - string);
- имя приложения, в котором создается роль (тип значения - string).

ПРИМЕР

Вызов: SecurityManagerApplication.CreateRole("role","app")

ChangeRole

```
void ChangeRole(string roleUID, string roleName)
```

Переименовывает роль из приложения, загруженного в элемент.

Входные параметры:

- уникальный идентификатор (uid) роли вида alpha:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx (тип значения - string);
- новое имя роли (тип значения - string).

ПРИМЕР

Вызов: SecurityManagerApplication.ChangeRole("alpha:4ce9483e-e137-41c2-bf1c-343e9b731f20","newrolename")

DeleteRole

```
void DeleteRole(string roleUID)
```

Удаляет роль из приложения, загруженного в элемент.

Входной параметр - уникальный идентификатор (uid) роли вида alpha:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx (тип значения - string).

ПРИМЕР

Вызов: SecurityManagerApplication.DeleteRole("alpha:4ce9483e-e137-41c2-bf1c-343e9b731f20")

RoleAddRight

```
void RoleAddRight(string appName, string roleUID, string rightName, uint1 rightType, string allowedValue, string forbidenValue)
```

Предоставляет для указанной роли право приложения, загруженного в элемент.

Входные параметры:

- имя приложения (тип значения - string);
- уникальный идентификатор (uid) роли вида alpha:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx (тип значения - string);
- имя права (тип значения - string);
- тип права (тип значения - uint1):
  - **«0»** - строковое;
  - **«1»** - логическое;
- разрешающее значение (тип значения - string);
- запрещающее значение (тип значения - string).

ПРИМЕР

Вызов: SecurityManagerApplication.RoleAddRight("appname","alpha:4ce9483e-e137-41c2-bf1c-343e9b731f20","pravo",0,"открыто","закрыто")

RoleChangeRight

```
void RoleChangeRight(string appName, string roleUID, string rightName, string allowedValue, string forbidenValue)
```

Меняет для указанной роли значения права приложения, загруженного в элемент.

Входные параметры:

- имя приложения (тип значения - string);
- уникальный идентификатор (uid) роли вида alpha:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx (тип значения - string);
- имя права (тип значения - string);
- новое разрешающее значение права (тип значения - string);
- новое запрещающее значение права (тип значения - string).

ПРИМЕР

Вызов: SecurityManagerApplication.RoleChangeRight("appname","alpha:4ce9483e-e137-41c2-bf1c-343e9b731f20","pravo","закрыто","открыто")

RoleDeleteRight

```
void RoleDeleteRight(string appName, string roleUID, string rightName)
```

Удаляет для указанной роли право приложения, загруженного в элемент.

Входные параметры:

- имя приложения (тип значения - string);
- уникальный идентификатор (uid) роли вида alpha:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx (тип значения - string);
- имя права (тип значения - string).

ПРИМЕР

Вызов: SecurityManagerApplication.RoleDeleteRight("appname","alpha:4ce9483e-e137-41c2-bf1c-343e9b731f20","pravo")

GetErrorDescriptionByCode

```
string GetErrorDescriptionByCode(uint1 FailReasonCode)
```

Возвращает текстовое описание ошибок, возникающих при загрузке и сохранении приложений.

Входной параметр - переменная FailReasonCode (тип значения - uint1).

2.7.3. События

LoadComplete

Информация о приложении загружена в элемент.

Активируется в случае успешного завершения операции Load() (стр. 88).

SaveComplete

Информация о приложении сохранена в подсистему безопасности SePlatform.Security.

Активируется в случае успешного завершения операции Save() (стр. 88).

LoadFailed

Информация о приложении не загружена в элемент.

Активируется в случае неуспешного завершения операции Load() (стр. 88).

Возвращает номер ошибки в переменную FailReasonCode, тип значения - uint1.

ПРИМЕЧАНИЕ

Получить текст ошибки можно, вызвав функцию GetErrorDescriptionByCode() (стр. 88).

SaveFailed

Информация о приложении не сохранена в подсистему безопасности SePlatform.Security.

Активируется в случае неуспешного завершения операции Save() (стр. 88).

Возвращает номер ошибки в переменную FailReasonCode, тип значения - uint1.

ПРИМЕЧАНИЕ

Получить текст ошибки можно, вызвав функцию GetErrorDescriptionByCode() (стр. 88).

2.8. Настройка безопасности: Пользователь (SecurityManagerUser)

Компонент предназначен для загрузки информации об учетной записи:

- имеющейся в подсистеме безопасности - для просмотра и изменения этой информации;
- созданной из проекта SePlatform.HMI - для дальнейшей отправки в подсистему безопасности.

ПРИМЕР

Сценарий применения свойств, функций и событий компонента приведен в разделе 1.5. Собственный конфигуратор подсистемы безопасности (стр. 22).

2.8.1. Свойства

Менеджер конфигурирования безопасности

Ссылка на элемент типа **Настройка безопасности: Менеджер**, обеспечивающий работу с учетными записями, группами пользователей и приложениями.

ОБРАТИТЕ ВНИМАНИЕ

Содержит свойство **Контекст безопасности** для указания ссылки на элемент типа **Контекст безопасности**.

Необходимо заполнить для взаимодействия с подсистемой безопасности SePlatform.Security.

UserID

Уникальный идентификатор (uid) учетной записи, загруженной в элемент. Тип значения - string.

Только для чтения в режиме рантайма.

ПРИМЕР

Вызов: SecurityManagerUser.UserID

Вид значения: alpha:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx

Пример значения: «**alpha:ab5cc552-a515-48ab-9e82-436aabf6bdfc**».

Login, Name, Surname, MidName, DisplayName, Position, Unit, Email, Phone, Comment

Свойства учетной записи, загруженной в элемент:

- логин;
- имя;
- фамилия;
- отчество;
- отображаемое имя;
- должность;
- подразделение;
- адрес электронной почты;
- номер телефона;
- комментарий.

Типы значений всех указанных свойств - string.

ПРИМЕР

Вызов: SecurityManagerUser.DisplayName

Пример значения: **«Иванов»**.

ForcePassChange

Требование сменены пароля при следующем входе. Тип значения - bool.

ПРИМЕР

Вызов: SecurityManagerUser.ForcePassChange

Значение:

- **«true»** - требовать смену пароля;
- **«false»** - не требовать смену пароля.

Disabled (Пользователь заблокирован)

Хранит информацию о том, заблокирован ли пользователь. Тип значения - bool.

ПРИМЕР

Вызов: SecurityManagerUser.Disabled

Значение:

- **«true»** - пользователь заблокирован;
- **«false»** - пользователь не заблокирован.

IsChanged

Уведомление об изменении в учетной записи, загруженной в элемент. Тип значения - bool.

Только для чтения в режиме рантайма.

ПРИМЕР

Вызов: SecurityManagerUser.IsChanged

Значение:

- **«true»** - учетная запись была изменена;
- **«false»** - нет изменений в учетной записи.

ПРИМЕЧАНИЕ

Принимает значение **«false»** при:

- загрузке учетной записи в элемент;
- сохранении учетной записи в подсистему безопасности.

2.8.2. Функции

Load

```
void Load(string userUID)
```

Загружает информацию об учетной записи из подсистемы безопасности в элемент.

Входной параметр - уникальный идентификатор (uid) учетной записи вида alpha:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx (тип значения - string).

ПРИМЕР

Вызов: SecurityManagerUser.Load("alpha:ab5cc552-a515-48ab-9e82-436aabf6bdfc")

Результат:

- в случае успешного завершения операции активируется событие LoadComplete (стр. 107);
- в случае неуспешного завершения операции активируется событие LoadFailed (стр. 107).

Save

```
void Save()
```

Отправляет информацию об учетной записи в подсистему безопасности после:

- редактирования ранее выгруженной из подсистемы безопасности учетной записи;
- создания учетной записи из проекта SePlatform.HMI с помощью функции New() и свойств элемента.

Функция не требует входных параметров.

ПРИМЕР

Вызов: SecurityManagerUser.Save()

Результат:

- в случае успешного завершения операции активируется событие SaveComplete (стр. 107);
- в случае неуспешного завершения операции активируется событие SaveFailed (стр. 107).

SetPassword

```
uint2 SetPassword(string password)
```

Устанавливает пароль для учетной записи, загруженной в элемент.

Входной параметр - пароль в виде строки. Функция возвращает код ошибки установки пароля. Коды ошибок аналогичны приведенным в описании функции ValidatePassword() (стр. 100).

ПРИМЕЧАНИЕ

Подсистема безопасности SePlatform.Security не накладывает требований к содержанию пароля. Требования к паролю устанавливаются путем назначения прав из стандартного приложения **`SePlatform.Security`**.

ValidatePassword

```
uint2 ValidatePassword(string password)
```

Проверяет устанавливаемый пароль на соответствие назначенным парольным политикам, без установки пароля пользователю.

Входной параметр - пароль в виде строки. Функция возвращает код несоответствия парольным политикам. Если функция возвращает **«0»**, ошибок нет.

Коды ошибок, возвращаемых функцией

Возвращаемое значение	Описание ошибки
1	Не используется.
2	Длина пароля не соответствует требуемой.
4	В пароле отсутствуют цифры.
8	В пароле отсутствуют символы верхнего регистра.
16	В пароле отсутствуют символы нижнего регистра.
32	В пароле отсутствуют специальные символы.
64	Ошибка при чтении значений прав, устанавливающих сложность пароля для пользователя.
128	Пароль должен отличаться от предыдущих паролей.
256	Пароль не должен содержать пробелы.
512	Ошибка при чтении значения права, устанавливающего минимальную длину пароля.

Возвращаемое значение	Описание ошибки
1024	Пароль не может быть пустым.

При возникновении нескольких ошибок, их коды складываются. Например, получено значение **«18»**. Это сумма **«2»** и **«16»**, где **«2»** - недопустимая длина пароля, а **«16»** - отсутствие символов нижнего регистра.

ПРИМЕЧАНИЕ

Подсистема безопасности SePlatform.Security не накладывает требований к содержанию пароля. Требования к паролю устанавливаются путем назначения прав из стандартного приложения **`SePlatform.Security`**.

GetGroupsList

```
string GetGroupsList()
```

Предоставляет список групп, в которых состоит пользователь учетной записи, загруженной в элемент.

Функция не требует входных параметров.

Возвращает данные в JSON-формате.

ПРИМЕР

Вызов: SecurityManagerUser.GetGroupsList()

Результат запроса списка групп в JSON-формате:

```
{
 "data": [
  {
   "groupID": "alpha:ab5cc552-a515-48ab-9e82-436aabf6bdfc",
   "groupName": "Диспетчеры"
  }
 ]
}
```

GetApplicationsList

```
string GetApplicationsList()
```

Предоставляет список приложений с правами пользователя, чья учетная запись загружена в элемент.

Функция не требует входных параметров.

Возвращает данные в JSON-формате.

ПРИМЕР

Вызов: SecurityManagerUser.GetApplicationsList()

Результат запроса списка приложений в JSON-формате:

```
{
 "data": [
  {
   "appID": "SePlatform.Security",
   "appName": "SePlatform.Security"
  },
  {
   "appID": "Управление состоянием оборудования",
   "appName": "Управление состоянием оборудования"
  }
 ]
}
```

GetRoles

```
string GetRoles(string appID)
```

Предоставляет список ролей пользователя, чья учетная запись загружена в элемент, в указанном приложении.

Входной параметр - ID приложения, из которого требуется получить список ролей (тип значения - string).

Возвращает данные в JSON-формате.

ПРИМЕР

Вызов: SecurityManagerUser.GetRoles("Управление состоянием оборудования")

Результат запроса списка приложений в JSON-формате:

```
{
 "data": [
  {
   "roleID": "alpha:ec5b145e-7d43-4c18-934c-1da6d4cb407f",
   "roleName": "rolename"
  }
 ]
}
```

GetRights

```
string GetRights(string appID)
```

Предоставляет список личных прав пользователя, чья учетная запись загружена в элемент, из указанного приложения.

Входной параметр - ID приложения, из которого требуется получить список прав (тип значения - string).

Возвращает данные в JSON-формате.

ОБРАТИТЕ ВНИМАНИЕ

Для получения списка эффективных прав используйте функцию GetEffectiveRights (стр. 103).

ПРИМЕР

Вызов: SecurityManagerUser.GetRights("Управление состоянием оборудования")

Результат запроса списка приложений в JSON-формате:

```
{
 "data": [
  {
   "appName": "Управление состоянием оборудования",
   "tokenType": "bool",
   "tokenName": "Управление резервуаром",
   "tokenDescription": "Право управления резервуаром",
   "ownerId": "alpha:07dba445-1903-498c-abdf-af7f55c145c4",
   "tokenValueAllow": "TRUE",
   "tokenValueDeny": "FALSE"
  }
 ]
}
```

GetEffectiveRights

```
string GetEffectiveRights(string appID)
```

Предоставляет список эффективных прав пользователя, чья учетная запись загружена в элемент, из указанного приложения.

Входной параметр - ID приложения, из которого требуется получить список эффективных прав (тип значения - string).

Возвращает данные в JSON-формате.

ПРИМЕР

Вызов: SecurityManagerUser.GetEffectiveRights("Управление состоянием оборудования")

Результат запроса списка приложений в JSON-формате:


```
{
 "data": [
  {
   "appName": "Управление состоянием оборудования",
   "tokenType": "bool",
   "tokenName": "Управление резервуаром",
   "tokenDescription": "Право управления резервуаром",
   "ownerId": "alpha:07dba445-1903-498c-abdf-af7f55c145c4",
   "tokenValueAllow": "TRUE",
   "tokenValueDeny": "FALSE"
  }
 ]
}
```


New

```
void New()
```

Освобождает элемент от загруженной в него информации для создания нового приложения.

Функция не требует входных параметров.

ПРИМЕР

Вызов: SecurityManagerUser.New()

ПРИМЕЧАНИЕ

После создания новой учетной записи не забудьте загрузить её в подсистему безопасности с помощью функции Save().

AddRole

```
void AddRole(string roleUID)
```

Назначает указанную роль пользователю, чья учетная запись загружена в элемент.

Входной параметр - уникальный идентификатор (uid) роли вида alpha:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx (тип значения - string).

ПРИМЕР

Вызов: SecurityManagerUser.AddRole("alpha:4ce9483e-e137-41c2-bf1c-343e9b731f20")

DeleteRole

```
void DeleteRole(string roleUID)
```

Лишает указанной роли пользователя, чья учетная запись загружена в элемент.

Входной параметр - уникальный идентификатор (uid) роли вида alpha:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx (тип значения - string).

ПРИМЕР

Вызов: SecurityManagerUser.DeleteRole("alpha:4ce9483e-e137-41c2-bf1c-343e9b731f20")

AddGroup

```
void AddGroup(string groupUID)
```

Добавляет в указанную группу пользователя, чья учетная запись загружена в элемент.

Входной параметр - уникальный идентификатор (uid) группы вида alpha:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx (тип значения - string).

ПРИМЕР

Вызов: SecurityManagerUser.AddGroup("alpha:ffdad1f9-ae36-4980-82b8-c7180a63b451")

DeleteGroup

```
void DeleteGroup(string groupUID)
```

Удаляет из указанной группы пользователя, чья учетная запись загружена в элемент.

Входной параметр - уникальный идентификатор (uid) группы вида alpha:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx (тип значения - string).

ПРИМЕР

Вызов: SecurityManagerUser.DeleteGroup("alpha:ffdad1f9-ae36-4980-82b8-c7180a63b451")

AddRight

```
void AddRight(string appID, string rightName, uint1 rightType, string allowedValue, string
forbidenValue)
```

Назначает право пользователю, чья учетная запись загружена в элемент.

Входные параметры:

- ID приложения, содержащего право (тип значения - string);
- имя права (тип значения - string);
- тип права (тип значения - uint1):
  - **«0»** - строковый;
  - **«1»** - логический;
- разрешающее значение права (тип значения - string);
- запрещающее значение права (тип значения - string).

ПРИМЕР

Вызов: SecurityManagerUser.AddRight("Управление состоянием оборудования","Управление насосом",1,"true","false")

ChangeRight

```
void ChangeRight(string appID, string rightName, string allowedValue, string forbidenValue)
```

Меняет значения прав пользователя, чья учетная запись загружена в элемент.

Входные параметры:

- ID приложения, содержащего право (тип значения - string);
- имя права (тип значения - string);
- новое разрешающее значение права (тип значения - string);
- новое запрещающее значение права (тип значения - string).

ПРИМЕР

Вызов: SecurityManagerUser.ChangeRight("Управление состоянием оборудования","Управление насосом","false","true")

DeleteRight

```
void DeleteRight(string appID, string rightName)
```

Лишает права пользователя, чья учетная запись загружена в элемент.

Входные параметры:

- ID приложения, содержащего право (тип значения - string);
- имя права (тип значения - string).

ПРИМЕР

Вызов: SecurityManagerUser.DeleteRight("Управление состоянием оборудования","Управление насосом")

GetErrorDescriptionByCode

```
string GetErrorDescriptionByCode(uint1 FailReasonCode)
```

Возвращает текстовое описание ошибок, возникающих при загрузке и сохранении учетных записей.

Входной параметр - переменная FailReasonCode (тип значения - uint1).

2.8.3. События

LoadComplete

Информация об учетной записи загружена в элемент.

Активируется в случае успешного завершения операции Load() (стр. 99).

SaveComplete

Информация об учетной записи сохранена в подсистему безопасности SePlatform.Security.

Активируется в случае успешного завершения операции Save() (стр. 99).

LoadFailed

Информация об учетной записи не загружена в элемент.

Активируется в случае неуспешного завершения операции Load() (стр. 99).

Возвращает номер ошибки в переменную FailReasonCode, тип значения - uint1.

ПРИМЕЧАНИЕ

Получить текст ошибки можно, вызвав функцию GetErrorDescriptionByCode() (стр. 99).

SaveFailed

Информация об учетной записи не сохранена в подсистему безопасности SePlatform.Security.

Активируется в случае неуспешного завершения операции Save() (стр. 99).

Возвращает номер ошибки в переменную FailReasonCode, тип значения - uint1.

ПРИМЕЧАНИЕ

Получить текст ошибки можно, вызвав функцию GetErrorDescriptionByCode() (стр. 99).

2.9. Настройка безопасности: Группа (SecurityManagerGroup)

Компонент предназначен для загрузки информации о группе:

- имеющейся в подсистеме безопасности - для просмотра и изменения этой информации;
- созданной из проекта SePlatform.HMI - для дальнейшей отправки в подсистему безопасности.

2.9.1. Свойства

Менеджер конфигурирования безопасности

Ссылка на элемент типа **Настройка безопасности: Менеджер**, обеспечивающий работу с учетными записями, группами пользователей и приложениями.

ОБРАТИТЕ ВНИМАНИЕ

Содержит свойство Контекст безопасности для указания ссылки на элемент типа **Контекст безопасности**.

Необходимо заполнить для взаимодействия с подсистемой безопасности SePlatform.Security.

GroupId

Уникальный идентификатор (uid) группы, загруженной в элемент. Тип значения - string.

Только для чтения в режиме рантайма.

ПРИМЕР

Вызов: SecurityManagerGroup.GroupID

Вид значения: alpha:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx

Пример значения: **«alpha:d01969e0-ae05-4548-9375-2533b326a588»**.

ПРИМЕЧАНИЕ

Уникальный идентификатор формируется подсистемой безопасности в момент создания группы.

Имя группы (GroupName)

Название группы, загруженной в элемент. Тип значения - string.

ПРИМЕР

Вызов: SecurityManagerGroup.GroupName

Пример значения: **«Диспетчеры»**.

Описание группы (GroupDescription)

Описание группы, загруженной в элемент. Тип значения - string.

ПРИМЕР

Вызов: SecurityManagerGroup.GroupDescription

Disabled (Группа заблокирована)

Хранит информацию о том, заблокирована ли группа. Тип значения - bool.

ПРИМЕР

Вызов: SecurityManagerGroup.Disabled

Значение:

- **«true»** - группа заблокирована;
- **«false»** - группа не заблокирована.

IsChanged

Уведомление об изменении в группе, загруженной в элемент. Тип значения - bool.

Только для чтения в режиме рантайма.

ПРИМЕР

Вызов: SecurityManagerGroup.IsChanged

Значение:

- **«true»** - группа была изменена;
- **«false»** - нет изменений в группе.

ПРИМЕЧАНИЕ

Принимает значение **«false»** при:

- загрузке группы в элемент;
- сохранении группы в подсистему безопасности.

2.9.2. Функции

Load

```
void Load(string uid)
```

Загружает информацию о группе из подсистемы безопасности в элемент.

Входной параметр - уникальный идентификатор (uid) группы вида alpha:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx (тип значения - string).

ПРИМЕР

Вызов: SecurityManagerGroup.Load("alpha:d01969e0-ae05-4548-9375-2533b326a588")

Результат:

- в случае успешного завершения операции активируется событие LoadComplete (стр. 116);
- в случае неуспешного завершения операции активируется событие LoadFailed (стр. 116).

Save

```
void Save()
```

Отправляет информацию о группе в подсистему безопасности после:

- редактирования ранее выгруженной из подсистемы безопасности группы;
- создания группы из проекта SePlatform.HMI с помощью функции New() и свойств элемента.

Функция не требует входных параметров.

ПРИМЕР

Вызов: SecurityManagerGroup.Save()

Результат:

- в случае успешного завершения операции активируется событие SaveComplete (стр. 116);
- в случае неуспешного завершения операции активируется событие SaveFailed (стр. 116).

GetMembersList

```
string GetMembersList()
```

Предоставляет список участников группы, загруженной в элемент. Участниками группы могут быть как пользователи, так и дочерние группы.

Функция не требует входных параметров.

Возвращает данные в JSON-формате, где:

- memberID - уникальный идентификатор (uid) участника группы вида alpha:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx;
- memberName - логин участника группы;
- memberType - значение этого атрибута указывает, является участник пользователем или дочерней группой:
  - **«0»** - пользователь;
  - **«1»** - дочерняя группа.

ПРИМЕР

Вызов: SecurityManagerGroup.GetMembersList()

Результат запроса списка групп в JSON-формате:

```
{
 "data": [
  {
   "memberID": "alpha:3fbc8d1f-8384-4b19-933b-0d4f0dc59c10",
   "memberName": "ivanov",
   "memberType": "0"
  },
  {
   "memberID": "alpha:dd13c739-1387-4f06-8087-cb437f577def",
   "memberName": "krylov",
   "memberType": "0"
  }
 ]
}
```

GetApplicationsList

```
string GetApplicationsList()
```

Предоставляет список приложений с правами группы, загруженной в элемент.

Функция не требует входных параметров.

Возвращает данные в JSON-формате.

ПРИМЕР

Вызов: SecurityManagerGroup.GetApplicationsList()

Результат запроса списка групп в JSON-формате:

```
{
 "data": [
  {
   "appID": "Управление состоянием оборудования",
   "appName": "Управление состоянием оборудования"
  }
 ]
}
```

GroupRoles

```
string GroupRoles(string appID)
```

Предоставляет список ролей группы, загруженной в элемент, в указанном приложении.

Входной параметр - ID приложения, из которого требуется получить список ролей (тип значения - string).

Возвращает данные в JSON-формате.

ПРИМЕР

Вызов: SecurityManagerGroup.GroupRoles("Управление состоянием оборудования")

Результат запроса списка приложений в JSON-формате:

```
{
 "data": [
  {
   "roleID": "alpha:ec5b145e-7d43-4c18-934c-1da6d4cb407f",
   "roleName": "rolename"
  }
 ]
}
```

GroupRights

```
string GroupRights(string appID)
```

Предоставляет список прав группы, загруженной в элемент, из указанного приложения.

Входной параметр - ID приложения, из которого требуется получить список прав (тип значения - string).

Возвращает данные в JSON-формате.

ОБРАТИТЕ ВНИМАНИЕ

Для получения списка эффективных прав используйте функцию GroupEffectiveRights (стр. 113).

ПРИМЕР

Вызов: SecurityManagerGroup.GroupRights("Управление состоянием оборудования")

Результат запроса списка приложений в JSON-формате:

```
{
 "data": [
  {
   "appName": "Управление состоянием оборудования",
   "tokenType": "bool",
   "tokenName": "Управление резервуаром",
   "tokenDescription": "Право управления резервуаром",
   "ownerId": "alpha:07dba445-1903-498c-abdf-af7f55c145c4",
   "tokenValueAllow": "TRUE",
   "tokenValueDeny": "FALSE"
  }
 ]
}
```

GroupEffectiveRights

```
string GroupEffectiveRights(string appID)
```

Предоставляет список эффективных прав группы, загруженной в элемент, из указанного приложения.

Входной параметр - ID приложения, из которого требуется получить список эффективных прав (тип значения - string).

Возвращает данные в JSON-формате.

ПРИМЕР

Вызов: SecurityManagerGroup.GroupEffectiveRights("Управление состоянием оборудования")

Результат запроса списка приложений в JSON-формате:

```
{
 "data": [
  {
   "appName": "Управление состоянием оборудования",
   "tokenType": "bool",
   "tokenName": "Управление резервуаром",
   "tokenDescription": "Право управления резервуаром",
   "ownerId": "alpha:07dba445-1903-498c-abdf-af7f55c145c4",
   "tokenValueAllow": "TRUE",
   "tokenValueDeny": "FALSE"
  }
 ]
}
```

New

```
void New()
```

Освобождает элемент от загруженной в него информации для создания новой группы.

Функция не требует входных параметров.

ПРИМЕР

Вызов: SecurityManagerGroup.New()

ПРИМЕЧАНИЕ

После создания новой группы не забудьте загрузить её в подсистему безопасности с помощью функции Save().

AddRole

```
void AddRole(string roleUID)
```

Назначает указанную роль группе, загруженной в элемент.

Входной параметр - уникальный идентификатор (uid) роли вида alpha:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx (тип значения - string).

ПРИМЕР

Вызов: SecurityManagerGroup.AddRole("alpha:4ce9483e-e137-41c2-bf1c-343e9b731f20")

DeleteRole

```
void DeleteRole(string roleUID)
```

Лишает указанной роли группу, загруженную в элемент.

Входной параметр - уникальный идентификатор (uid) роли вида alpha:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx (тип значения - string).

ПРИМЕР

Вызов: SecurityManagerGroup.DeleteRole("alpha:4ce9483e-e137-41c2-bf1c-343e9b731f20")

AddRight

```
void AddRight(string appID, string rightName, uint1 rightType, string allowedValue, string
forbidenValue)
```

Назначает право группе, загруженной в элемент.

Входные параметры:

- ID приложения, содержащего право (тип значения - string);
- имя права (тип значения - string);
- тип права (тип значения - uint1):
  - **«0»** - строковый;
  - **«1»** - логический;
- разрешающее значение права (тип значения - string);
- запрещающее значение права (тип значения - string).

ПРИМЕР

Вызов: SecurityManagerGroup.AddRight("Управление состоянием оборудования","Управление насосом",1,"true","false"

ChangeRight

```
void ChangeRight(string appID, string rightName, string allowedValue, string forbidenValue)
```

Меняет значения прав группы, загруженной в элемент.

Входные параметры:

- ID приложения, содержащего право (тип значения - string);
- имя права (тип значения - string);
- новое разрешающее значение права (тип значения - string);
- новое запрещающее значение права (тип значения - string).

ПРИМЕР

Вызов: SecurityManagerGroup.ChangeRight("Управление состоянием оборудования","Управление насосом","false","true")

DeleteRight

```
void DeleteRight(string appID, string rightName)
```

Лишает права группу, загруженную в элемент.

Входные параметры:

- ID приложения, содержащего право (тип значения - string);
- имя права (тип значения - string).

ПРИМЕР

Вызов: SecurityManagerGroup.DeleteRight("Управление состоянием оборудования","Управление насосом")

GetErrorDescriptionByCode

```
string GetErrorDescriptionByCode(uint1 FailReasonCode)
```

Возвращает текстовое описание ошибок, возникающих при загрузке и сохранении групп.

Входной параметр - переменная FailReasonCode (тип значения - uint1).

2.9.3. События

LoadComplete

Информация о группе загружена в элемент.

Активируется в случае успешного завершения операции Load() (стр. 109).

SaveComplete

Информация о группе сохранена в подсистему безопасности SePlatform.Security.

Активируется в случае успешного завершения операции Save() (стр. 109).

LoadFailed

Информация о группе не загружена в элемент.

Активируется в случае неуспешного завершения операции Load() (стр. 109).

Возвращает номер ошибки в переменную FailReasonCode, тип значения - uint1.

ПРИМЕЧАНИЕ

Получить текст ошибки можно, вызвав функцию GetErrorDescriptionByCode() (стр. 109).

SaveFailed

Информация о группе не сохранена в подсистему безопасности SePlatform.Security.

Активируется в случае неуспешного завершения операции Save() (стр. 109).

Возвращает номер ошибки в переменную FailReasonCode, тип значения - uint1.

ПРИМЕЧАНИЕ

Получить текст ошибки можно, вызвав функцию GetErrorDescriptionByCode() (стр. 109).

2.10. Мастер конфигурирования Security (Configurator)

Компонент предназначен для чтения и обновления конфигурации службы Агент SePlatform.Security.

Обычно сразу после установки подсистемы безопасности SePlatform.Security необходимо настроить ее компоненты - в том числе, службу Агент SePlatform.Security. Настройка службы выполняется в конфигурационном файле `seplatform.security.agent.xml`, расположенном в `C:\Program Files\SePlatform\SePlatform.Security` (для Windows) или в `/opt/SePlatform/SePlatform.Security` (для Linux). Подробнее настройка службы описана в документе на SePlatform.Security, в разделе **«Настройка компонентов»**. Вносить изменения в конфигурационный файл можно вручную, либо из проекта SePlatform.HMI с помощью свойств, функций и событий экземпляра **Мастер конфигурирования Security**. Компонент следует использовать следующим образом:

- Для чтения конфигурации необходимо вызвать функцию Read() (стр. 122). Она поместит данные из конфигурационного файла в компонент. После этого можно будет обратиться к данным, прочитав значения свойств компонента или вызвав функции, название которых начинается с **Get** (**GetLdapHost()**, **GetLdapPort()** и пр.).
- Для создания новой конфигурации необходимо сначала указать новые данные, поместив их в значения свойств и во входные параметры функций, название которых начинается с **Add** (**AddLdap()**, **AddLogConsumer()** и пр.). После этого следует вызвать функцию Generate() (стр. 122), которая создаст текст нового конфигурационного файла.

Пример использования компонента в проекте SePlatform.HMI приведен в разделе 1.7. Конфигурирование агента безопасности (стр. 41).

2.10.1. Свойства

Адрес Net агента (NetHost)

IP-адрес или имя компьютера, на котором установлен Net-агент, к которому подключается агент безопасности. Под Net-агентом подразумевается служба **SePlatform.Net.Agent** (на ОС Windows) или **SePlatform.Net.service** (на ОС Linux). Тип значения - string.

Зачастую Net-агент установлен на локальном компьютере.

Значение свойства соответствует значению атрибута `Address` элемента `<EntryPointNetAgent>` в конфигурационном файле агента безопасности.

Порт Net агента (NetPort)

Номер порта для подключения к Net-агенту, к которому подключается агент безопасности. Тип значения - int4.

Значение свойства соответствует значению атрибута `Port` элемента `<EntryPointNetAgent>` в конфигурационном файле агента безопасности.

Адрес LDAP сервера (LdapHost)

IP-адрес или имя компьютера, на котором развернут LDAP-сервер, к которому подключается агент безопасности. Тип значения - string.

Свойство устарело. Теперь при настройке агента может быть указано несколько LDAP-серверов, поэтому значения адресов записываются в компонент в виде массива.

Чтобы прочитать конкретное значение в массиве:

1. Вызовите функцию Read() (стр. 122), предназначенную для записи текущей конфигурации агента в массив.
2. Вызовите функцию GetLdapHost() (стр. 123), указав в качестве входного параметра номер элемента в массиве.

Количество элементов в массиве (LDAP-серверов) записывается в значение свойства LdapCount (стр. 121). Каждый элемент массива соответствует значению атрибута `Address` одного из элементов `<LDAPServer>` в конфигурационном файле агента безопасности.

Чтобы добавить описание нового LDAP-сервера, используйте функцию AddLdap() (стр. 123)

Порт LDAP сервера (LdapPort)

Номер порта для подключения к LDAP-серверу. Тип значения - string.

Свойство устарело. Теперь при настройке агента может быть указано несколько LDAP-серверов, поэтому значения портов записываются в компонент в виде массива.

Чтобы прочитать конкретное значение в массиве:

1. Вызовите функцию Read() (стр. 122), предназначенную для записи текущей конфигурации агента в массив.
2. Вызовите функцию GetLdapPort() (стр. 124), указав в качестве входного параметра номер элемента в массиве.

Количество элементов в массиве (LDAP-серверов) записывается в значение свойства LdapCount (стр. 121). Каждый элемент массива соответствует значению атрибута `Port` одного из элементов `<LDAPServer>` в конфигурационном файле агента безопасности.

Чтобы добавить описание нового LDAP-сервера, используйте функцию AddLdap() (стр. 123)

Использование защищенного соединения (UseSecureConnection)

Указывает агенту безопасности необходимость использования защищенного соединения с LDAP-сервером. Тип значения - bool.

Значение свойства соответствует значению атрибута `value` элемента `<LdapSecure>` в конфигурационном файле агента безопасности.

Режим работы контроля целостности (ICMode)

Указывает, включен ли режим контроля целостности файлов и папок. Тип значения - bool.

Значение свойства соответствует значению атрибута `ICMode` элемента `<Options>` в конфигурационном файле агента безопасности:

- **«0»** - контроль целостности отключен, значение свойства - **«fasle»**;
- **«1»** - контроль целостности включен, значение свойства - **«true»**.

Пользователь LDAP (LdapUser)

Учетная запись администратора LDAP-сервера. Тип значения - string.

Поскольку учетная запись хранится на LDAP-сервере в виде каталога, обращение к ней происходит в стандартном для LDAP виде:

```
cn=ЛогинПользователя,dc=ДоменLDAPсервера
```

Значение свойства соответствует значению атрибута `value` элемента `<LdapUser>` в конфигурационном файле агента безопасности.

ПРИМЕР

Пример значения: **«cn=Manager,dc=maxcrc,dc=com»**

Пароль пользователя LDAP (LdapUserPass)

Пароль учетной записи администратора LDAP-сервера. Тип значения - string.

Поскольку пароли являются секретной информацией, просмотреть текущее значение при чтении конфигурации нельзя. При создании новой конфигурации свойство используется для записи нового значения. Значение пароля, введенное в открытом виде, будет зашифровано.

Адрес папки системы безопасности (LdapDN)

Название корневого каталога на LDAP-сервере. Тип значения - string.

Обращение к каталогу происходит в стандартном для LDAP виде:

```
ou=НазваниеПапки,dc=ДоменLDAPсервера
```

Значение свойства соответствует значению атрибута `value` элемента `<SecurityDn>` в конфигурационном файле агента безопасности.

ПРИМЕР

Пример значения: **«cn=ou=SePlatformSecurity,dc=maxcrc,dc=com»**

Имя гостевой УЗ (GuestName)

Имя гостевой учетной записи - записи, чьи права используются, когда нет активной пользовательской сессии. Тип значения - string.

Значение свойства соответствует значению атрибута `value` элемента `<GuestDisplayName>` в конфигурационном файле агента безопасности.

Пользователь по умолчанию (DefaultUser)

Имя учетной записи пользователя по умолчанию - пользователя, чья сессия становится активной, если другие пользователи не авторизовались в подсистеме. Тип значения - string.

Значение свойства соответствует значению атрибута `value` элемента `<DefaultUser>` в конфигурационном файле агента безопасности.

Пароль пользователя по умолчанию (DefaultUserPass)

Пароль пользователя по умолчанию. Тип значения - string.

Значение свойства соответствует значению атрибута `value` элемента `<DefaultUserPassword>` в конфигурационном файле агента безопасности.

Уровень логирования (LoggerLevel)

Означает количество информации, выводимой в лог подсистемы безопасности. Тип значения - int4.

Значение свойства соответствует значению атрибута `LoggerLevel` элемента `<Options>` в конфигурационном файле агента безопасности:

- **«0»** - в лог выводится минимум информации;
- **«2»** - в лог выводится вся основная информация о работе SePlatform.Security;
- **«5»** - в лог выводится дополнительная информация о работе SePlatform.Security помимо основной.

Комбинации блокировки (DriverString)

Перечень клавиш, заблокированных для использования пользователем. Тип значения - string.

Заблокированные сочетания указаны в виде SCAN-кодов клавиш, разделенных внутри сочетания символом **«+»**, а между сочетаниями - символом **«;»**.

Значение свойства соответствует значению атрибута `kbDriverString` элемента `<Options>` в конфигурационном файле агента безопасности.

ПРИМЕР

Пример значения: **«0x1D+0x38+0x53;0x1D+0x2A+0x01;»**

В данном примере заблокированы сочетания **«ctrl+alt+del»** и **«ctrl+shift+esc»**.

Трассировка аудита (TraceAudit)

Включает трансляцию сообщений аудита в системный журнал. Тип значения - bool.

Значение свойства соответствует значению атрибута `TraceAudit` элемента `<AuditLogConsumers>` в конфигурационном файле агента безопасности:

- **«0»** - сообщения не транслируются в системный журнал, значение свойства - **«fasle»**;
- **«1»** - сообщения транслируются в системный журнал, значение свойства - **«true»**.

Кэш прав пользователя (UseRightsStorageCache)

Указывает, используются ли кэшированные значения прав пользователя. Тип значения - bool.

Значение свойства соответствует значению атрибута `UseRightsCacheStorage` элемента `<Options>` в конфигурационном файле агента безопасности:

- **«0»** - актуальные значения прав пользователя запрашиваются с LDAP-сервера постоянно, значение свойства - **«false»**;

› **«1»** - значения прав пользователя запрашиваются с LDAP-сервера только в момент входа пользователя в подсистему, а в процессе его работы значения запрашиваются из кэша. Значение свойства в таком случае- **«true»**.

Префикс сообщений аудита (MesPrefix)

Префикс сообщений аудита. Тип значения - string.

Значение свойства соответствует значению атрибута `value` элемента `<mesPrefix>` в конфигурационном файле агента безопасности.

Ошибка конфигурирования (Error)

Текст ошибки, возникшей при попытке создания новой конфигурации Агент SePlatform.Security с помощью функции Generate() (стр. 122). Тип значения - string.

Только для чтения в режиме рантайма.

Сгенерированная строка (GeneratedString)

Полный текст конфигурационного файла Агент SePlatform.Security в xml-формате, созданного в результате вызова функции Generate() (стр. 122). Тип значения - string.

Только для чтения в режиме рантайма.

Ошибка чтения (ReadError)

Текст ошибки, возникшей при попытке чтения конфигурации Агент SePlatform.Security с помощью функции Read() (стр. 122). Тип значения - string.

Только для чтения в режиме рантайма.

Количество LDAP серверов (LdapCount)

Количество LDAP-серверов, указанных в конфигурационном файле Агент SePlatform.Security. Тип значения - int4.

Только для чтения в режиме рантайма.

Количество потребителей аудита (ConsumersCount)

Количество серверов-потребителей аудита, указанных в конфигурационном файле Агент SePlatform.Security. Тип значения - int4.

Только для чтения в режиме рантайма.

Незаполненные поля (UndefinedListItems)

Список полей (свойств), для которых обязательно указать значение для создания новой конфигурации Агент SePlatform.Security с помощью функции Generate() (стр. 122). Тип значения - string.

Только для чтения в режиме рантайма.

2.10.2. Функции

Generate

```
int4 Generate(bool activationFlag)
```

Создает текст конфигурационного файла Агент SePlatform.Security.

Входной параметр отвечает за совместимость создаваемой конфигурации со старыми версиями SePlatform.Security (тип значения - bool). В качестве значения следует указывать **«true»**.

Результат:

- сразу после вызова функции активируется событие GenerationStarted (стр. 133);
- в случае успешного завершения операции активируется событие GenerationFinished (стр. 134), а строка-результат записывается в свойство GeneratedString (стр. 121);
- в случае неуспешного завершения операции активируется событие GenerationFailure (стр. 134).

Для создания конфигурации должны быть заполнены все обязательные поля. Обязательно следует указать:

- адрес и порт Net-агента - в соответствующих свойствах (стр. 117);
- параметры хотя бы одного LDAP-сервера - с помощью функции AddLdap() (стр. 123);
- учетную запись администратора LDAP (в стандартном для LDAP виде) и пароль (стр. 118);
- пароль пользователя, чья учетная запись используется по умолчанию (стр. 120);
- адрес корневой папки LDAP-сервера (стр. 119);
- защищено ли соединение агента безопасности с LDAP-сервером (стр. 118).

Если в процессе создания конфигурации возникнет ошибка, функция вернет ее код. Чтобы ознакомиться с текстом ошибки, обратитесь ко значению свойства Ошибка конфигурирования (Error) (стр. 121) компонента, либо ко значению внутренней переменной Error события GenerationFailure (стр. 134).

Read

```
void Read(string XML)
```

Выполняет чтение конфигурации из конфигурационного файла Агент SePlatform.Security.

Входной параметр - строка, содержащая полный текст конфигурационного файла Агент SePlatform.Security в xml-формате (тип значения - string).

Чтобы получить такую строку из файла с помощью компонентов SePlatform.HMI, используйте функцию ReadTextFile() компонента **Окружение: файлы (File System)**. В качестве входного параметра функции укажите полный путь к файлу конфигурации.

ПРИМЕР

Пример вызова:

```
Configurator.Read(FileSystem.ReadTextFile("C:/Program
Files/SePlatform/SePlatform.Security/seplatform.security.agent.xml"));
```

Результат:

- сразу после вызова функции активируется событие ReadingStarted (стр. 134);
- в случае успешного завершения операции активируется событие ReadingFinished (стр. 134);
- в случае неуспешного завершения операции активируется событие ReadingFailure (стр. 134).

AddLdap

```
bool AddLdap(string LDAPHost, int4 LDAPPort)
```

Добавляет описание LDAP-сервера, к которому должен подключаться агент безопасности. Добавленное описание хранится во внутреннем массиве компонента.

Входные параметры:

- IP-адрес или имя компьютера, на котором развернут LDAP-сервер;
- номер порта для подключения к LDAP-серверу.

Функция возвращает результат добавления LDAP-сервера в конфигурацию:

- **«true»** - информация добавлена;
- **«false»** - информацию не удалось добавить.

В случае успешного завершения операции активируется событие LdapListChanged (стр. 134).

ClearLdapList

```
void ClearLdapList()
```

Очищает внутренний массив LDAP-серверов компонента.

Функция не требует входных параметров.

В случае успешного завершения операции активируется событие LdapListChanged (стр. 134).

GetLdapHost

```
string GetLdapHost(int4 index)
```

Предоставляет IP-адрес одного из указанных в конфигурационном файле LDAP-серверов.

Поскольку в конфигурационном файле может быть указано несколько LDAP-серверов, перечень их IP-адресов записывается в компонент в виде массива. Обратиться к конкретному значению можно по номеру его записи в массив.
Каждый элемент массива соответствует значению атрибута `Address` одного из xml-элементов `<LDAPServer>`, вложенного в xml-элемент `<LdapHosts>`, в конфигурационном файле агента безопасности.

GetLdapPort

```
int4 GetLdapPort(int4 index)
```

Предоставляет номер порта для подключения к одному из указанных в конфигурационном файле LDAP-серверов.

Поскольку в конфигурационном файле может быть указано несколько LDAP-серверов, перечень портов для подключения к ним записывается в компонент в виде массива. Обратиться к конкретному значению можно по номеру его записи в массив.

Каждый элемент массива соответствует значению атрибута `Port` одного из xml-элементов `<LDAPServer>`, вложенного в xml-элемент `<LdapHosts>`, в конфигурационном файле агента безопасности.

GetAuditServerHost

```
string GetAuditServerHost(int4 index)
```

Предоставляет IP-адрес одного из указанных в конфигурационном файле серверов-потребителей аудита.

Поскольку в конфигурационном файле может быть указано несколько серверов-потребителей, перечень их IP-адресов записывается в компонент в виде массива. Обратиться к конкретному значению можно по номеру его записи в массив.

Каждый элемент массива соответствует значению атрибута `Host` xml-элемента `<Server>`, вложенного в один из xml-элементов `<OpcDaLogConsumer>`, в конфигурационном файле агента безопасности.

GetAuditServerPort

```
int4 GetAuditServerPort(int4 index)
```

Предоставляет порт для подключения к одному из указанных в конфигурационном файле серверов-потребителей аудита.

Поскольку в конфигурационном файле может быть указано несколько серверов-потребителей, перечень портов для подключения к ним записывается в компонент в виде массива. Обратиться к конкретному значению можно по номеру его записи в массив.

Каждый элемент массива соответствует значению атрибута `TCPServerPort` xml-элемента `<Server>`, вложенного в один из xml-элементов `<OpcDaLogConsumer>`, в конфигурационном файле агента безопасности.

GetServerProgId

```
string GetAuditServerPort(int4 index)
```

Предоставляет ProgID (строковый идентификатор) одного из указанных в конфигурационном файле серверов-потребителей аудита.

Поскольку в конфигурационном файле может быть указано несколько серверов-потребителей, перечень их ProgID записывается в компонент в виде массива. Обратиться к конкретному значению можно по номеру его записи в массив.

Каждый элемент массива соответствует значению атрибута `ProgId` xml-элемента `<Server>`, вложенного в один из xml-элементов `<OpcDaLogConsumer>`, в конфигурационном файле агента безопасности.

GetServerType

```
string GetServerType(int4 index)
```

Предоставляет информацию о типе одного из указанных в конфигурационном файле серверов-потребителей аудита.

Поскольку в конфигурационном файле может быть указано несколько серверов-потребителей, перечень их типов записывается в компонент в виде массива. Обратиться к конкретному значению можно по номеру его записи в массив.

Каждый элемент массива соответствует значению атрибута `Type` xml-элемента `<Server>`, вложенного в один из xml-элементов `<OpcDaLogConsumer>`, в конфигурационном файле агента безопасности.

GetSeverityCount

```
int4 GetServerType(int4 index)
```

Предоставляет количество категорий важности сообщений для одного из указанных в конфигурационном файле серверов-потребителей аудита.

Поскольку в конфигурационном файле может быть указано несколько серверов-потребителей, значение от каждого из них записывается в компонент в виде элемента массива. Обратиться к конкретному значению можно по номеру его записи в массив.

Каждый элемент массива соответствует количеству элементов `<Severity>` внутри элемента `<SeverityMap>`, вложенного в один из элементов `<OpcDaLogConsumer>`, в конфигурационном файле агента безопасности.

GetSignalsCount

```
int4 GetSignalsCount(int4 index)
```

Предоставляет количество сигналов, предназначенных для записи сообщений аудита, в одном из указанных в конфигурационном файле серверов-потребителей аудита.

Поскольку в конфигурационном файле может быть указано несколько серверов-потребителей, значение для каждого из них записывается в компонент в виде элемента массива. Обратиться к конкретному значению можно по номеру его записи в массив.

Каждый элемент массива соответствует количеству элементов `<Signal>` внутри элемента `<SignalMap>`, вложенного в один из элементов `<OpcDaLogConsumer>`, в конфигурационном файле агента безопасности.

GetSeverityCategory

```
string GetSeverityCategory(int4 i, int4 k)
```

Предоставляет название выбранной категории важности сообщений.

Поскольку в конфигурационном файле может быть указано несколько серверов-потребителей, каждый из них описывается в компоненте в виде элемента массива A[i]. Каждый элемент массива A[i] представляет собой массив B, каждый элемент B[i,k] которого описывает одну из категорий важности описываемого сервера A[i].

```
Массив A, описывающий сервера-потребители = [[Массив B, описывающий первый сервер], [Массив
B, описывающий второй сервер]]
Массив B, описывающий первый сервер = [[Описание первой категории важности], [Описание
второй категории важности]]
Массив B, описывающий второй сервер = [[Описание первой категории важности], [Описание
второй категории важности]]
```

Обратиться к конкретной категории важности можно по номеру элемента в массивах A и B (i, k).

Каждый элемент массива B[i,k] соответствует одному из значений атрибута `Category` xml-элемента `<Severity>`, вложенного в xml-элемент `<SeverityMap>` одного из xml-элементов `<OpcDaLogConsumer>` в конфигурационном файле агента безопасности.

Пример использования приведен в описании функции GetSeverityValue().

GetSeverityValue

```
int4 GetSeverityValue(int4 i, int4 k)
```

Предоставляет числовое значение выбранной категории важности сообщений.

Поскольку в конфигурационном файле может быть указано несколько серверов-потребителей, каждый из них описывается в компоненте в виде элемента массива A[i]. Каждый элемент массива A[i] представляет собой массив B, каждый элемент B[i,k] которого описывает одну из категорий важности описываемого сервера A[i].

```
Массив A, описывающий сервера-потребители = [[Массив B, описывающий первый сервер], [Массив
B, описывающий второй сервер]]
```

```
Массив B, описывающий первый сервер = [[Описание первой категории важности], [Описание
второй категории важности]]
Массив B, описывающий второй сервер = [[Описание первой категории важности], [Описание
второй категории важности]]
```

Обратиться к конкретной категории важности можно по номеру элемента в массивах A и B (i, k).

Каждый элемент массива B[i,k] соответствует одному из значений атрибута `Value` xml-элемента `<Severity>`, вложенного в xml-элемент `<SeverityMap>` одного из xml-элементов `<OpcDaLogConsumer>` в конфигурационном файле агента безопасности.

ПРИМЕР

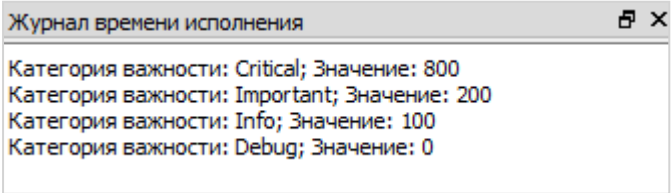
Допустим, в конфигурационном файле описан один сервер-потребитель сообщений (один элемент <OpcDaLogConsumer>), для которого описано четыре категории важности сообщений аудита:

```
<AuditLogConsumers TraceAudit="1">
 <OpcDaLogConsumer>
  <Server Host="127.0.0.1" Type="OPC" ...>
   <SeverityMap>
    <Severity Category="Critical" Value="800"/>
    <Severity Category="Important" Value="200"/>
    <Severity Category="Info" Value="100"/>
    <Severity Category="Debug" Value="0"/>
   </SeverityMap>
  </Server>
 </OpcDaLogConsumer>
</AuditLogConsumers>
```

Чтобы получить список категорий важности сообщений с их значениями в каждом сервере-потребителе, вызовите нужные функции в коде, выполняющемся в случае успешного чтения конфигурации Агент SePlatform.Security (например, в обработчике события ReadingFinished (стр. 134)). Укажите в качестве входных параметров индексы **«i»** (индекс в массиве серверов-потребителей) и **«k»** (индекс в массиве категорий важности). Приведенный ниже пример написан на языке SePlatform.Om, в нем итоговый список записывается в лог:

```
i: int4 = 0;
while (i < Configurator.ConsumersCount) //цикл выполняется, пока в массиве A не будут
описаны все сервера-потребители
{
  k: int4 = 0;
  while (k < Configurator.GetSeverityCount(i)) //цикл выполняется, пока в массив B не
будут записаны все категории важности i-го сервера-потребителя
  {
    DebugTool.Log("Категория важности: "+Configurator.GetSeverityCategory(i,k)+";
Значение: "+String.ToString(Configurator.GetSeverityValue(i,k))); //в Журнал времени
исполнения записываются названия и значения категорий важности для i-го сервера
    k += 1;
  }
  i += 1;
}
```

В результате вызова функций в **Журнал времени исполнения** запишутся названия категорий важности с их значениями:



GetSignalName

```
string GetSignalName(int4 i, int4 k)
```

Предоставляет название указанного сигнала, предназначенного для записи сообщений аудита, в указанном сервере-потребителе.

Поскольку в конфигурационном файле может быть указано несколько серверов-потребителей, каждый из них описывается в компоненте в виде элемента массива A[i]. Каждый элемент массива A[i] представляет собой массив B, каждый элемент B[i,k] которого описывает один из сигналов в описываемом сервере A[i].

```
Массив A, описывающий сервера-потребители = [[Массив B, описывающий первый сервер], [Массив
B, описывающий второй сервер]]
Массив B, описывающий первый сервер = [[Описание первого сигнала], [Описание второго
сигнала]]
Массив B, описывающий второй сервер = [[Описание первого сигнала], [Описание второго
сигнала]]
```

Обратиться к конкретному сигналу можно по номеру элемента в массивах A и B (i, k).

Каждый элемент массива B[i,k] соответствует одному из значений атрибута `Name` xml-элемента `<Signal>`, вложенного в xml-элемент `<SignalMap>` одного из xml-элементов `<OpcDaLogConsumer>` в конфигурационном файле агента безопасности.

Пример использования приведен в описании функции GetSignalType().

GetSignalMode

```
int4 GetSignalMode(int4 i, int4 k)
```

Предоставляет информацию о режиме записи сообщения в указанный сигнал в указанном сервере-потребителе. Возвращает значение в числовом виде, где:

- **«1»** - соответствует **«DynamicEvent»** (запись сообщения с xml-конструкцией, описывающей динамическое событие);
- **«2»** - соответствует **«Value»** (обычная запись сообщения).

Поскольку в конфигурационном файле может быть указано несколько серверов-потребителей, каждый из них описывается в компоненте в виде элемента массива A[i]. Каждый элемент массива A[i] представляет собой массив B, каждый элемент B[i,k] которого описывает один из сигналов в описываемом сервере A[i].

```
Массив A, описывающий сервера-потребители = [[Массив B, описывающий первый сервер], [Массив
B, описывающий второй сервер]]
Массив B, описывающий первый сервер = [[Описание первого сигнала], [Описание второго
сигнала]]
Массив B, описывающий второй сервер = [[Описание первого сигнала], [Описание второго
сигнала]]
```

Обратиться к конкретному сигналу можно по номеру элемента в массивах A и B (i, k).

Каждый элемент массива B[i,k] соответствует одному из значений атрибута `Mode` xml-элемента `<Signal>`, вложенного в xml-элемент `<SignalMap>` одного из xml-элементов `<OpcDaLogConsumer>` в конфигурационном файле агента безопасности.
Пример использования приведен в описании функции GetSignalType().

GetSignalType

```
string GetSignalType(int4 i, int4 k)
```

Предоставляет название типа сообщений в указанном сигнале для указанного сервера-потребителя.

Поскольку в конфигурационном файле может быть указано несколько серверов-потребителей, каждый из них описывается в компоненте в виде элемента массива A[i]. Каждый элемент массива A[i] представляет собой массив B, каждый элемент B[i,k] которого описывает один из сигналов в описываемом сервере A[i].

```
Массив A, описывающий сервера-потребители = [[Массив B, описывающий первый сервер], [Массив
B, описывающий второй сервер]]
Массив B, описывающий первый сервер = [[Описание первого сигнала], [Описание второго
сигнала]]
Массив B, описывающий второй сервер = [[Описание первого сигнала], [Описание второго
сигнала]]
```

Обратиться к конкретному сигналу можно по номеру элемента в массивах A и B (i, k).

Каждый элемент массива B[i,k] соответствует одному из значений атрибута `Type` xml-элемента `<Signal>`, вложенного в xml-элемент `<SignalMap>` одного из xml-элементов `<OpcDaLogConsumer>` в конфигурационном файле агента безопасности.

ПРИМЕР

Допустим, в конфигурационном файле описан один сервер-потребитель сообщений (один элемент <OpcDaLogConsumer>), в котором для записи сообщений аудита предназначено несколько сигналов:

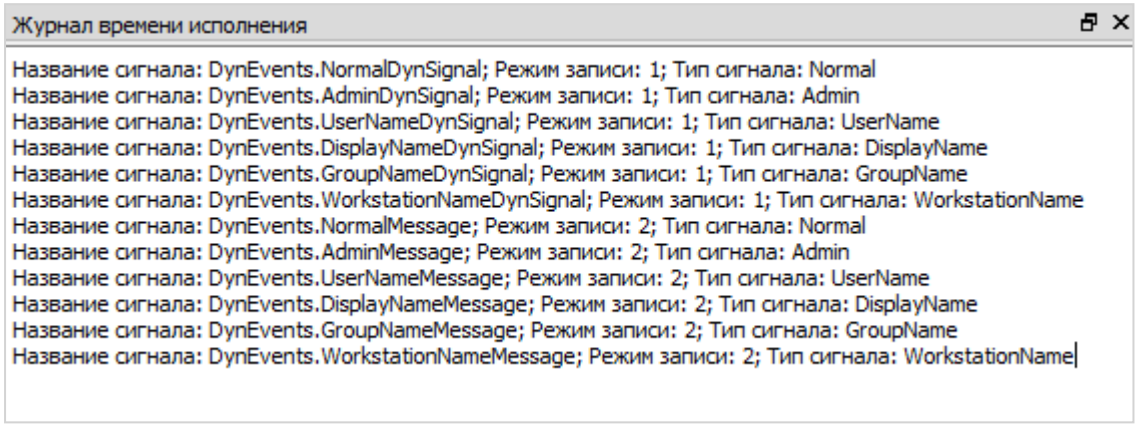
```
<AuditLogConsumers TraceAudit="1">
 <OpcDaLogConsumer>
  <Server Host="127.0.0.1" Type="OPC" ...>
   <SignalMap>
    <Signal Name="DynEvents.NormalDynSignal" Mode="DynamicEvent" Type="Normal"/>
    <Signal Name="DynEvents.AdminDynSignal" Mode="DynamicEvent" Type="Admin"/>
    <Signal Name="DynEvents.UserNameDynSignal" Mode="DynamicEvent" Type="UserName"/>
    <Signal Name="DynEvents.DisplayNameDynSignal" Mode="DynamicEvent"
Type="DisplayName"/>
    <Signal Name="DynEvents.GroupNameDynSignal" Mode="DynamicEvent"
Type="GroupName"/>
    <Signal Name="DynEvents.WorkstationNameDynSignal" Mode="DynamicEvent"
Type="WorkstationName"/>
    <Signal Name="DynEvents.NormalMessage" Mode="Value" Type="Normal"/>
    <Signal Name="DynEvents.AdminMessage" Mode="Value" Type="Admin"/>
    <Signal Name="DynEvents.UserNameMessage" Mode="Value" Type="UserName"/>
    <Signal Name="DynEvents.DisplayNameMessage" Mode="Value" Type="DisplayName"/>
    <Signal Name="DynEvents.GroupNameMessage" Mode="Value" Type="GroupName"/>
    <Signal Name="DynEvents.WorkstationNameMessage" Mode="Value"
Type="WorkstationName"/>
   </SignalMap>
  </Server>
 </OpcDaLogConsumer>
</AuditLogConsumers>
```

Чтобы получить список параметров сигналов в каждом сервере-потребителе, вызовите нужные функции в коде, выполняющемся в случае успешного чтения конфигурации Агент SePlatform.Security (например, в обработчике события ReadingFinished (стр. 134)). Укажите в качестве входных параметров индексы **«i»** (индекс в массиве серверов-потребителей) и **«k»** (индекс в массиве сигналов). Приведенный ниже пример написан на языке SePlatform.Om, в нем список сигналов с их параметрами записывается в лог:

```
i: int4 = 0;
while (i < Configurator.ConsumersCount) //цикл выполняется, пока в массиве A не будут
описаны все сервера-потребители
{
  k: int4 = 0;
  while (k < Configurator.GetSignalsCount(i)) //цикл выполняется, пока в массив B не
будут записаны все сигналы i-го сервера-потребителя
  {
    DebugTool.Log("Название сигнала: "+Configurator.GetSignalName(i,k)+"; Режим
записи: "+String.ToString(Configurator.GetSignalMode(i,k))+"; Тип сообщения:
"+Configurator.GetSignalType(i,k)); //в Журнал времени исполнения записывается список
сигналов i-го сервера
    k += 1;
```

```
    }
    i += 1;
  }
```

В результате вызова функций в **Журнал времени исполнения** запишется список сигналов с их параметрами:



ClearLogConsumersList

```
void ClearLogConsumersList()
```

Очищает внутренний массив серверов-потребителей в компоненте.

Функция не требует входных параметров.

В случае успешного завершения операции активируется событие ConsumersListChanged (стр. 134).

AddLogConsumer

```
void AddLogConsumer(string Host, string HostTcpReserve, int4 TCPServerPort, string ProgID,
string Type)
```

Добавляет описание сервера-потребителя аудита сообщений. Добавленное описание хранится во внутреннем массиве компонента.

Входные параметры:

- имя или IP-адрес сервера;
- имя или IP-адрес сервера для резервного канала связи с сервером TCP (значение может быть пустым);
- порт для подключения к TCP-серверу (если выбран TCP-сервер);
- строковый идентификатор OPC-сервера (если выбран OPC-сервер);
- тип сервера:
  - для Windows возможны значения **«OPC»** (по умолчанию) и **«TCP»**;
  - для других ОС допустимо только **«TCP»**.

В случае успешного завершения операции активируется событие ConsumersListChanged (стр. 134).

AddSeverity

```
bool AddSeverity(int4 consumerIndex, string severityCategory, int4 severityValue)
```

Добавляет в компонент категорию важности сообщений для указанного сервера-потребителя.

Входные параметры:

- индекс сервера-потребителя во внутреннем массиве компонента;
- название категории важности;
- значение категории важности.

Функция возвращает результат добавления категории в компонент:

- **«true»** - категория добавлена;
- **«false»** - категорию не удалось добавить. В таком случае следует проверить значение индекса - оно может быть больше, чем размер массива в компоненте.

AddSignal

```
bool AddSignal(int4 consumerIndex, string signalName, int4 signalMode, string signalType)
```

Добавляет в компонент информацию о сигнале, предназначенном для записи сообщений, для указанного сервера-потребителя.

Входные параметры:

- индекс сервера-потребителя во внутреннем массиве компонента;
- название сигнала;
- режим записи сообщения в сигнал:
  - **«1»** - соответствует **«DynamicEvent»** (запись сообщения с xml-конструкцией, описывающей динамическое событие);
  - **«2»** - соответствует **«Value»** (обычная запись сообщения);
- тип сообщений, записывающихся в сигнал.

Функция возвращает результат добавления информации о сигнале в компонент:

- **«true»** - информация добавлена;
- **«false»** - информацию не удалось добавить. В таком случае следует проверить значение индекса - оно может быть больше, чем размер массива в компоненте.

2.10.3. События

GenerationStarted

Начато создание текста конфигурационного файла Агент SePlatform.Security.

Активируется сразу после вызова функции Generate() (стр. 122).

GenerationFinished

Создан текст конфигурационного файла Агент SePlatform.Security. Результат помещен в свойство GeneratedString (стр. 121)

Активируется в результате успешного выполнения функции Generate() (стр. 122).

GenerationFailure

Не удалось создать текст конфигурационного файла Агент SePlatform.Security.

Активируется в результате неуспешного выполнения функции Generate() (стр. 122).

Чтобы ознакомиться с текстом ошибки, обратитесь ко значению внутренней переменной Error этого события, либо ко значению свойства Ошибка конфигурирования (Error) (стр. 121) компонента.

ReadingStarted

Начато чтение конфигурационного файла Агент SePlatform.Security.

Активируется сразу после вызова функции Read() (стр. 122).

ReadingFinished

Чтение конфигурационного файла Агент SePlatform.Security завершено успешно.

Активируется в результате успешного выполнения функции Read() (стр. 122).

Чтобы ознакомиться с полученными настройками, обратитесь ко значениям свойств компонента.

ReadingFailure

Не удалось прочитать конфигурационный файл Агент SePlatform.Security.

Активируется в результате неуспешного выполнения функции Read() (стр. 122).

Чтобы ознакомиться с текстом ошибки, обратитесь ко значению внутренней переменной Error этого события, либо ко значению свойства Ошибка чтения (ReadError) (стр. 121) компонента.

LdapListChanged

Изменен список LDAP-серверов во внутреннем массиве компонента.

Активируется в результате успешного выполнения одной из функций – AddLdap() или ClearLdapList() (стр. 123).

ConsumersListChanged

Изменен список серверов-потребителей сообщений аудита во внутреннем массиве компонента.

Активируется в результате успешного выполнения одной из функций - AddLogConsumer() или ClearLogConsumersList() (стр. 132).

2.11. Информация лицензирования: Получение (LicenseInfo)

Компонент предназначен для получения информации о лицензии на компоненты Систэм Платформ (далее - продукты), установленные на локальной или удаленных рабочих станциях, включенных в сеть SePlatform.Net.

Чтобы получить информацию о лицензии из проекта SePlatform.HMI, необходимо:

1. Добавить экземпляр **`LicenseInfo`** на форму.
2. Отправить JSON-запрос серверу лицензирования SePlatform.License Server, вызвав функцию компонента **RequestLicenseInfo()** (или **RequestRemoteLicenseInfo()**, если запрос отправляется на удаленную рабочую станцию). В качестве входного параметра функции указать JSON-запрос. Шаблоны JSON-запросов приведены ниже.
3. В обработчике событий **RequestLicenseInfoComplete** (**RequestRemoteLicenseInfoComplete**) указать, что должно происходить при успешном получении JSON-ответа. Для чтения ответа можно использовать внутреннюю переменную события LicenseInfoJSON, куда записывается полученный JSON-ответ. Также ответ записывается в свойство **LicenseInfo** компонента. Примеры JSON-ответов приведены ниже.

   ПРИМЕЧАНИЕ

   При отправке нескольких JSON-запросов из одной команды следует учитывать разницу в чтении ответа из внутренней переменной LicenseInfoJSON и из свойства **LicenseInfo**.

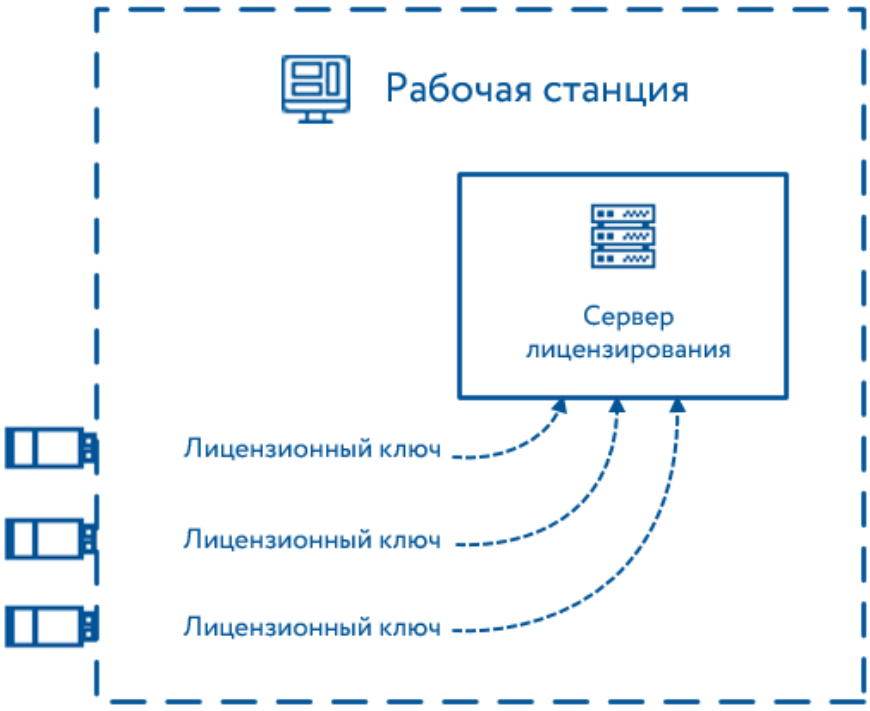
   Значение свойства **LicenseInfo** перезаписывается при получении каждого нового значения, а в переменной LicenseInfoJSON полученное значение сохраняется до передачи в обработчик события. Так как процесс обработки асинхронный, то к моменту выполнения обработчика значение свойства **LicenseInfo** может несколько раз измениться. А значение аргумента LicenseInfoJSON будет тем же, что и в момент наступления события.
4. В обработчике событий **RequestLicenseInfoFailed** (**RequestRemoteLicenseInfoFailed**) указать, что должно происходить, если JSON-ответ от сервера получить не удалось. У события есть внутренняя переменная FailReasonCode, куда записывается номер ошибки. Чтобы получить текст ошибки по ее номеру, используйте функцию **GetErrorDescriptionByCode()** компонента.

ОБРАТИТЕ ВНИМАНИЕ

Компонент может запрашивать информацию только у сервера лицензирования SePlatform.License Server версии 1.2.2 и новее.

Какую информацию предоставляет сервер лицензирования

Сервер лицензирования - это сервер, хранящий всю информацию о правах на использование продуктов Систэм Платформ и/или их отдельных функций. В дальнейшем такие права будем называть лицензиями. Лицензии предоставляются лицензионными ключами. Сервер лицензирования объединяет информацию о лицензиях, предоставляемых всеми лицензионными ключами. На каждой рабочей станции устанавливается свой сервер лицензирования.



ПРИМЕЧАНИЕ

Ключи бывают аппаратными и программными. Для лицензирования продуктов используются два вида лицензионных ключей: Sentinel и Guardant. Подробнее об этом - в документе, описывающем лицензирование Систэм Платформ.

Таким образом, сервер лицензирования может предоставить:

- информацию об отдельном лицензионном ключе - его типе, о предоставляемых им лицензиях на компоненты и/или их функции;
- информацию о конкретной лицензии на продукт и/или его функцию - на определенном ключе или на устройстве в целом (со всех ключей);
- лицензионные данные - общую информацию обо всех лицензиях на продукты и/или их функции - на определенном ключе или на устройстве в целом (со всех ключей);
- полную лицензионную информацию - лицензионные данные с устройства в целом (со всех ключей).

Как построить JSON-запрос к серверу лицензирования

Чтобы построить JSON-запрос к серверу лицензирования, необходимо указать команду запроса и объект, которого касается запрос. Некоторые команды требуют указания параметров. Параметры для каждой команды описаны в подразделах с примерами запросов, приведенном ниже.

```
{
 "command": "команда",
 "object": "объект",
 "params": {}
}
```

Объект запроса - это описанная выше информация, которую предоставляет сервер лицензирования:

- Лицензионный ключ `licenseKeys`. При запросе информации о лицензионном ключе в качестве объекта запроса укажите **«licenseApi/licenseInfo/licenseKeys»**.
- Лицензия на продукт и/или его функцию `licenseFeature`. При обращении к лицензии на продукт/функцию в качестве объекта запроса укажите **«licenseApi/licenseInfo/licenseFeature»**.
- Лицензионные данные `licenseData`. При обращении к лицензионным данным в качестве объекта запроса укажите **«licenseApi/licenseInfo/licenseData»**.
- Лицензионная информация `licenseInfo`. Включает в себя лицензионные данные `licenseData` и информацию о лицензионных ключах `licenseKeys`. При обращении к полной лицензионной информации в качестве объекта запроса укажите **«licenseApi/licenseInfo/licenseInfo»**.

В запросе можно указать одну из следующих команд:

- `Get` - команда для получения информации о конкретном объекте. При использовании этой команды необходимо указывать идентификатор объекта, которого касается запрос. Команду можно отправить объектам `licenseKeys` и `licenseFeature`.
- `List` - команда для получения списка объектов конкретного типа. В ответе будут получены только значимые поля. Команду можно отправить объектам `licenseKeys`, `licenseData` и `licenseInfo`.
- `Fetch` - команда для получения списка объектов конкретного типа. В ответе будет получена вся информация об указанных объектах. Команду можно отправить объектам `licenseKeys`, `licenseData` и `licenseInfo`.

Примеры JSON-запросов и возвращаемых JSON-ответов

Ниже приведены примеры запросов к серверу лицензирования и его ответов для каждого объекта.

ОБРАТИТЕ ВНИМАНИЕ

При построении запроса кавычки внутри него необходимо экранировать обратным слешем.

licenseKeys - Get: запрос информации о конкретном ключе

Чтобы получить информацию о конкретном лицензионном ключе, в параметрах запроса необходимо указать идентификатор ключа `licenseKeyId` и его класс `licenseKeyClass`.

Пример запроса:

```
{
 "command": "Get",
 "object": "licenseApi/licenseInfo/licenseKeys",
 "params":
  {
   "licenseKeyId":"0123456789",
   "licenseKeyClass":"Guardant"
  }
}
```

Пример ответа:

```
{
 "result":
```

```
 {
  "licenseKeyId":"0123456789",
  "licenseKeyClass":"Guardant",
  "licenseKeyType":"Guardant Sign",
  "licenseData":
  [
   {
    "licenseFeatureId":"11",
    "licenseFeatureDescription":"SePlatform.HMI",
    "licenseFeatureCountType":"PER_DOMAIN",
    "licenseFeatureCount":"1"
   },
   {
    "licenseFeatureId":"12",
    ...
  ]
 }
}
```

Описание полей JSON-ответа

Поле	Описание
`licenseKeyId`	Идентификатор ключа.
`licenseKeyClass`	Класс ключа. Примеры значений - `HASP`, `Guardant`.
`licenseKeyType`	Тип ключа. Обычно здесь указывается, программный это ключ или аппаратный.

`licenseData`	Массив информации обо всех лицензиях на продукты и/или функции, записанных на этот ключ. Здесь:
	› `licenseFeatureId` - идентификатор лицензии на продукт и/или функцию;
	› `licenseFeatureDescription` - название продукта или его функции;
	› `licenseFeatureCountType` - тип учёта количества лицензий. Возможные значения:
	  › `PER_PROCESS` - количество лицензий на процесс;
	  › `PER_HOST` - количество лицензий на хост;
	  › `PER_DOMAIN` - количество лицензий на домен.
	› `licenseFeatureCount` - количество лицензий, предоставляемых этим ключом.

licenseKeys - List: запрос краткой информации обо всех ключах

Под краткой информацией подразумевается перечень используемых ключей, включающий идентификаторы и названия классов ключей. Чтобы получить краткую информацию обо всех лицензионных ключах, установленных на рабочую станцию, обратитесь к объекту `licenseKeys` с командой `List`.

Пример запроса:

```
{
 "command": "List",
 "object": "licenseApi/licenseInfo/licenseKeys",
}
```

Пример ответа:

```
{
 "result":
 {
  "licenseKeys":
  [
   {
    "licenseKeyId":"0123456789",
    "licenseKeyClass":"Guardant"
   },
   {
    "licenseKeyId":"0123456788",
    "licenseKeyClass":"Guardant"
   }
  ]
 }
}
```

Описание полей JSON-ответа

Поле	Описание
`licenseKeys`	Массив информации о ключах. Здесь: › `licenseKeyId` - идентификатор ключа; › `licenseKeyClass` - класс ключа. Примеры значений - `HASP`, `Guardant`.

licenseKeys - Fetch: запрос полной информации обо всех ключах

Чтобы получить полную информацию обо всех лицензионных ключах, установленных на рабочей станции, обратитесь к объекту `licenseKeys` с командой `Fetch`.

Пример запроса:

```
{
    "command": "Fetch",
    "object": "licenseApi/licenseInfo/licenseKeys"
}
```

Пример ответа:

```
{
 "result":
 {
  "licenseKeys":
  [
   {
    "licenseKeyId":"0123456789",
    "licenseKeyClass":"Guardant",
    "licenseKeyType":"Guardant Sign",
    "licenseData":
    [
     {
      "licenseFeatureId":"11",
      "licenseFeatureDescription":"SePlatform.HMI",
      "licenseFeatureCountType":"PER_DOMAIN",
      "licenseFeatureCount":"1"
     },
     {
     "licenseFeatureId":"12",
     ...
    ]
   }
  ],
  [
   {
    "licenseKeyId":"0123456788",
    "licenseKeyClass":"Guardant",
    ...
   },
  ]
 }
}
```

Описание полей JSON-ответа

Поле	Описание
`licenseKeys`	Массив информации обо всех ключах, установленных на рабочей станции. Количество вложенных объектов массива (в квадратных скобках) соответствует количеству ключей.
`licenseKeyId`	Идентификатор ключа.
`licenseKeyClass`	Класс ключа. Примеры значений - `HASP`, `Guardant`.
`licenseKeyType`	Тип ключа. Обычно здесь указывается, программный это ключ или аппаратный.

Поле	Описание
`licenseData`	Массив информации обо всех лицензиях на продукты и/или функции, записанных на этот ключ. Здесь: › `licenseFeatureId` - идентификатор лицензии на продукт и/или функцию; › `licenseFeatureDescription` - название продукта или его функции; › `licenseFeatureCountType` - тип учёта количества лицензий. Возможные значения: › `PER_PROCESS` - количество лицензий на процесс; › `PER_HOST` - количество лицензий на хост; › `PER_DOMAIN` - количество лицензий на домен. › `licenseFeatureCount` - количество лицензий, предоставляемых этим ключом.

licenseFeature - Get: запрос информации о лицензии на продукт или функцию

Чтобы получить информацию о лицензировании конкретного продукта/функции, в параметрах запроса необходимо указать источник:

› если требуется информация с определенного лицензионного ключа, в параметрах указываются идентификатор ключа `licenseKeyId`, его класс `licenseKeyClass` и идентификатор продукта/функции `licenseFeatureId`.

› если требуется полная информация со всех ключей, в параметрах указывается только идентификатор продукта/функции `licenseFeatureId`.

Пример запроса с определенного лицензионного ключа:

```
{
 "command": "Get",
 "object": "licenseApi/licenseInfo/licenseFeature",
 "params":
  {
   "licenseKeyId":"0123456789",
   "licenseKeyClass":"Guardant",
   "licenseFeatureId":"11"
  }
}
```

Пример запроса со всех ключей:

```
{
 "command": "Get",
 "object": "licenseApi/licenseInfo/licenseFeature",
 "params":
  {
   "licenseFeatureId":"11"
  }
}
```

Пример ответа:

```
{
 "result":
 {
  "licenseFeatureId":"11",
  "licenseFeatureDescription":"SePlatform.HMI",
  "licenseFeatureCountType":"PER_DOMAIN",
  "licenseFeatureCount":"1"
 }
}
```

Описание полей JSON-ответа

Поле	Описание
`licenseFeatureId`	Идентификатор лицензии на продукт и/или функцию.
`licenseFeatureDescription`	Название продукта или функции.
`licenseFeatureCountType`	Тип учёта количества лицензий. Возможные значения: › `PER_PROCESS` - количество лицензий на процесс; › `PER_HOST` - количество лицензий на хост; › `PER_DOMAIN` - количество лицензий на домен.
`licenseFeatureCount`	Количество лицензий, предоставляемых указанным ключом или всеми ключами.

licenseData - List: запрос краткой информации обо всех лицензиях на продукты и/или функции

Под краткой информацией подразумевается перечень лицензий, включающий их идентификаторы и названия продуктов и/или функций. Чтобы получить краткую информацию обо всех лицензиях на продукты и функции, в параметрах запроса необходимо указать источник:

› если требуются лицензионные данные с определенного лицензионного ключа, в параметрах указываются идентификатор ключа `licenseKeyId` и его класс `licenseKeyClass`.

› если требуются лицензионные данные со всех ключей, параметры указывать не нужно. Обратите внимание, что результат запроса со всех ключей содержит информацию обо всех возможных лицензиях на продукты и функции - в том числе тех, которые могут быть не активны на вашем устройстве.

Пример запроса с определенного лицензионного ключа:

```
{
 "command": "List",
 "object": "licenseApi/licenseInfo/licenseData",
 "params":
  {
   "licenseKeyId":"0123456789",
   "licenseKeyClass":"Guardant"
```

```
  }
}
```

Пример запроса со всех ключей:

```
{
 "command": "List",
 "object": "licenseApi/licenseInfo/licenseData",
}
```

Пример ответа:

```
{
 "result":
 {
  "licenseData":
  [
   {
    "licenseFeatureId":"11",
    "licenseFeatureDescription":"SePlatform.HMI"
   },
   {
    "licenseFeatureId":"12",
    "licenseFeatureDescription":"SePlatform.HMI"
   }
   ...
  ]
 }
}
```

Описание полей JSON-ответа

Поле	Описание
`licenseFeatureId`	Идентификатор лицензии на продукт и/или функцию.
`licenseFeatureDescription`	Название продукта или функции.

licenseData - Fetch: запрос полной информации обо всех лицензиях на продукты и/или функции

Чтобы получить полную информацию обо всех лицензиях на продукты и функции, в параметрах запроса необходимо указать источник:

- если требуются лицензионные данные с определенного лицензионного ключа, в параметрах указываются идентификатор ключа `licenseKeyId` и его класс `licenseKeyClass`.
- если требуются лицензионные данные со всех ключей, параметры указывать не нужно. Обратите внимание, что результат запроса со всех ключей содержит информацию обо всех возможных лицензиях на продукты и функции - в том числе тех, которые могут быть не активны на вашем устройстве.

Пример запроса с определенного лицензионного ключа:

```
{
 "command": "Fetch",
 "object": "licenseApi/licenseInfo/licenseData",
 "params":
  {
   "licenseKeyId":"0123456789",
   "licenseKeyClass":"Guardant"
  }
}
```

Пример запроса со всех ключей:

```
{
 "command": "Fetch",
 "object": "licenseApi/licenseInfo/licenseData",
}
```

Пример ответа:

```
{
 "result":
 {
  "licenseData":
  [
   {
    "licenseFeatureId":"11",
    "licenseFeatureDescription":"SePlatform.HMI",
    "licenseFeatureCountType":"PER_DOMAIN",
    "licenseFeatureCount":"1"
   },
   {
    "licenseFeatureId":"12",
    "licenseFeatureDescription":"SePlatform.Server (1 000)",
    "licenseFeatureCountType":"PER_DOMAIN",
    "licenseFeatureCount":"0"
   }
   ...
  ]
 }
}
```

Описание полей JSON-ответа

Поле	Описание
`licenseFeatureId`	Идентификатор лицензии на продукт и/или функцию.
`licenseFeatureDescription`	Название продукта или функции.

Поле	Описание
`licenseFeatureCountType`	Тип учёта количества лицензий. Возможные значения: › `PER_PROCESS` - количество лицензий на процесс; › `PER_HOST` - количество лицензий на хост; › `PER_DOMAIN` - количество лицензий на домен.
`licenseFeatureCount`	Количество лицензий, предоставляемых указанным ключом или всеми ключами.

licenseInfo - List: запрос краткой лицензионной информации

Под краткой информацией подразумевается перечень ключей и предоставляемых ими лицензий. Чтобы получить краткую лицензионную информацию, обратитесь к объекту `licenseInfo` с командой `List`. Обратите внимание, что результат запроса содержит информацию обо всех возможных лицензиях на продукты и функции - в том числе тех, которые могут быть не активны на вашем устройстве.

Пример запроса:

```
{
 "command": "List",
 "object": "licenseApi/licenseInfo/licenseInfo"
}
```

Пример ответа:

```
{
 "result":
 {
  "licenseKeys":
  [
   {
    "licenseKeyId":"0123456789",
    "licenseKeyClass":"Guardant"
   },
   {
    "licenseKeyId":"0123456788",
    "licenseKeyClass":"Guardant"
   }
  ],
  "licenseData":
  [
   {
    "licenseFeatureId":"11",
    "licenseFeatureDescription":"SePlatform.HMI"
   },
   {
    "licenseFeatureId":"12",
    ...
```

```
   ]
  }
 }
```

Описание полей JSON-ответа

Поле	Описание
`licenseKeys`	Массив информации обо всех ключах, установленных на рабочей станции. Количество вложенных объектов массива (в квадратных скобках) соответствует количеству ключей. Здесь: › `licenseKeyId` - идентификатор ключа; › `licenseKeyClass` - класс ключа. Примеры значений - `HASP`, `Guardant`.
`licenseData`	Массив информации обо всех лицензиях на продукты и/или функции на сервере. Здесь: › `licenseFeatureId` - идентификатор лицензии на продукт и/или функцию; › `licenseFeatureDescription` - название продукта или его функции.

licenseInfo - Fetch: запрос полной лицензионной информации

Чтобы получить полную лицензионную информацию, обратитесь к объекту `licenseInfo` с командой `Fetch`. Обратите внимание, что результат запроса содержит информацию обо всех возможных лицензиях на продукты и функции - в том числе тех, которые могут быть не активны на вашем устройстве.

Пример запроса:

```
{
 "command": "Fetch",
 "object": "licenseApi/licenseInfo/licenseInfo"
}
```

Пример ответа:

```
{
 "result":
 {
  "licenseKeys":
  [
   {
    "licenseKeyId":"0123456789",
    "licenseKeyClass":"Guardant",
    "licenseKeyType":"Guardant Sign"
   },
   {
    "licenseKeyId":"0123456788",
    "licenseKeyClass":"Guardant",
    "licenseKeyType":"Guardant Sign"
   }
  ],
```

```
  "licenseData":
  [
   {
    "licenseFeatureId":"11",
    "licenseFeatureDescription":"SePlatform.HMI",
    "licenseFeatureCountType":"PER_DOMAIN",
    "licenseFeatureCount":"1234567"
   },
   {
    "licenseFeatureId":"12",
    "licenseFeatureDescription":"SePlatform.Server (1 000)",
    "licenseFeatureCountType":"PER_DOMAIN",
    "licenseFeatureCount":"0"
   }
   ...
  ]
 }
}
```


Описание полей JSON-ответа

Поле	Описание
`licenseKeys`	Массив информации обо всех ключах, установленных на рабочей станции. Количество вложенных объектов массива (в квадратных скобках) соответствует количеству ключей. Здесь: › `licenseKeyId` - идентификатор ключа; › `licenseKeyClass` - класс ключа. Примеры значений - `HASP`, `Guardant`; › `licenseKeyType` - тип ключа. Обычно здесь указывается, программный это ключ или аппаратный.
`licenseData`	Массив информации обо всех лицензиях на продукты и/или функции на сервере. Здесь: › `licenseFeatureId` - идентификатор лицензии на продукт и/или функцию; › `licenseFeatureDescription` - название продукта или его функции; › `licenseFeatureCountType` - тип учёта количества лицензий. Возможные значения: › `PER_PROCESS` - количество лицензий на процесс; › `PER_HOST` - количество лицензий на хост; › `PER_DOMAIN` - количество лицензий на домен. › `licenseFeatureCount` - количество лицензий, предоставляемых этим ключом.

2.11.1. Свойства

Контекст безопасности

Ссылка на элемент типа **Контекст безопасности**, обеспечивающий взаимодействие с подсистемой безопасности SePlatform.Security.

ОБРАТИТЕ ВНИМАНИЕ

Необходимо заполнить для взаимодействия с подсистемой безопасности SePlatform.Security.

LicenseInfo

Результат запроса информации о лицензии. Предоставляется в формате JSON. Тип значения - string.

Только для чтения в режиме рантайма.

Примеры результатов запросов приведены на странице описания компонента (стр. 135).

2.11.2. Функции

RequestLicenseInfo

```
void RequestLicenseInfo(string jsonRequest)
```

Выполняет запрос информации о лицензии на локальной рабочей станции.

Входной параметр - JSON-запрос. Примеры запросов приведены на странице описания компонента (стр. 135).

ОБРАТИТЕ ВНИМАНИЕ

Кавычки внутри JSON-запроса необходимо экранировать обратным слешем.

ПРИМЕР

Вызов: LicenseInfo.RequestLicenseInfo("
{\"command\":\"List\",\"object\":\"licenseApi/licenseInfo/licenseKeys\"}")

Результат:

- в случае успешного завершения операции активируется событие RequestLicenseInfoComplete (стр. 149);
- в случае неуспешного завершения операции активируется событие RequestLicenseInfoFailed (стр. 149).

RequestRemoteLicenseInfo

```
void RequestRemoteLicenseInfo(string jsonRequest, string netName)
```

Выполняет запрос информации о лицензии на удаленной рабочей станции.

Входные параметры:

- JSON-запрос;
- имя рабочей станции в сети SePlatform.Net.

ОБРАТИТЕ ВНИМАНИЕ

Кавычки внутри JSON-запроса необходимо экранировать обратным слешем.

ПРИМЕР

Вызов: LicenseInfo.RequestLicenseInfo("
{\"command\":\"List\",\"object\":\"licenseApi/licenseInfo/licenseKeys\"}", "Netname")

Результат:

- в случае успешного завершения операции активируется событие RequestRemoteLicenseInfoComplete (стр. 149);
- в случае неуспешного завершения операции активируется событие RequestRemoteLicenseInfoFailed (стр. 150).

GetErrorDescriptionByCode

```
string GetErrorDescriptionByCode(uint1 FailReasonCode)
```

Возвращает текстовое описание ошибок, возникающих при запросе информации о лицензии.

Входной параметр - переменная FailReasonCode (тип значения - uint1).

2.11.3. События

RequestLicenseInfoComplete

Получена информация о лицензии на локальной рабочей станции.

Активируется в случае успешного завершения операции RequestLicenseInfo() (стр. 148).

Возвращает данные в JSON-формате во внутреннюю переменную события LicenseInfoJSON, тип значения - string.

RequestRemoteLicenseInfoComplete

Получена информация о лицензии на удаленной рабочей станции.

Активируется в случае успешного завершения операции RequestRemoteLicenseInfo() (стр. 148).

Возвращает данные в JSON-формате во внутреннюю переменную события LicenseInfoJSON, тип значения - string.

RequestLicenseInfoFailed

Не удалось получить информацию о лицензии на локальной рабочей станции.

Активируется в случае неуспешного завершения операции RequestLicenseInfo() (стр. 148).

Возвращает номер ошибки во внутреннюю переменную события FailReasonCode, тип значения - uint1.

RequestRemoteLicenseInfoFailed

Не удалось получить информацию о лицензии на удаленной рабочей станции.

Активируется в случае неуспешного завершения операции RequestRemoteLicenseInfo() (стр. 148).

Возвращает номер ошибки во внутреннюю переменную события FailReasonCode, тип значения - uint1.

История изменений

2.0

Важно. Текущая версия предназначена для использования с SePlatform.HMI 2.0.

2.0.1

Улучшения

- Функция **RequestUserList()** компонента 2.6. Настройка безопасности: Менеджер (SecurityManager) (стр. 76), помимо прочих данных об учетных записях, теперь предоставляет их уникальные идентификаторы (uid).
- Функция **GetMembersList()** компонента 2.9. Настройка безопасности: Группа (SecurityManagerGroup) (стр. 108) теперь предоставляет больше данных. Помимо uid и логина (или названия) участника группы теперь предоставляется информация о том, является ли участник пользователем или дочерней группой.

Исправления

- Доработаны функции и события компонента 2.1. Контекст безопасности (SecurityContext) (стр. 47):
  - При обращении функции **RemoteGetLoggedUsersList()** к недоступной рабочей станции не активировалось событие **RemoteGetLoggedUsersFailed**.
  - События **ConnectedChanged** и **CurrentUserChanged** активировались дважды при запуске проекта SePlatform.HMI в рантайм, теперь - только один раз.
- Устранена причина, по которой значения логических прав не сбрасывались после выхода пользователя из подсистемы.

2.0.2

Функция **GetTokensList()** компонента **Настройка безопасности: Приложение** теперь предоставляет больше данных о правах приложений.

2.0.3

Обратите внимание: не рекомендуется использовать предыдущую версию 2.0.2 из-за иногда возникающих ошибок запуска проектов SePlatform.HMI, использующих компоненты безопасности.

Новая возможность

Разработан компонент **Информация лицензирования: Получение (LicenseInfo)**. Компонент предназначен для получения информации о лицензии на продукты Систэм Платформ, установленные на локальной или удаленных рабочих станциях, включенных в сеть SePlatform.Net.

Исправления

- Устранена причина, по которой недавно удаленная учетная запись появлялась в списке пользователей, полученном в результате вызова функции **RequestUsersList()** компонента **Настройка безопасности: Менеджер (SecurityManager)**.
- Метод проверки соответствия пароля требованиям **ValidatePassword()** компонента **Настройка безопасности: Пользователь (SecurityManagerUser)** теперь корректно работает и для кириллических паролей.

2.0.4

Исправления

- Исправлена ошибка, из-за которой был невозможен вход под учетной записью, которой требовалась смена пароля, во время другой активной пользовательской сессии.
- Устранена причина активации двух событий компонента **Настройка безопасности: Менеджер** при вызове функции ExportConfiguration() - успешное и неуспешное сохранение резервной копии конфигурации - если сохранить копию пытался пользователь без прав на создание файлов в указанной папке.
- Теперь при сохранении учетной записи пользователя с помощью функции Save() компонента **Настройка безопасности: Пользователь** выполняется проверка, не указан ли пустой пароль.
- Исправлена ошибка, из-за которой у пользователя не учитывались права, назначенные его группе из приложения, если этой группе также была назначена роль из этого приложения.
- Устранена причина, по которой функция GetApplicationsList() компонента **Настройка безопасности: Группа** не возвращала список прав группы, унаследованных от родительской группы.
- Исправлена ошибка, из-за которой у компонента **Настройка безопасности: Контроль целостности** активировались события об успешном завершении операции при обращении к несуществующему узлу.
- События ConnectedChanged и CurrentUserChanged компонента **Контекст безопасности** снова активируются дважды при запуске проекта SePlatform.HMI в рантайм. Два события - это более правильное поведение компонента. В первый раз событие активируется при начальной инициализации проекта. Второй раз - когда агент передает в проект информацию о текущем пользователе.
- Исправлены другие некритичные ошибки.

2.0.5

Исправление

Устранена причина, по которой функция GroupDisplayName() компонента **Контекст безопасности** не возвращала значений. Ошибка возникала при использовании предыдущей версии - 2.0.4.

2.0.7

Исправления

- Устранена причина, по которой свойство GuestMode компонента **Контекст безопасности** предоставляло некорректное значение при первом запуске проекта в рантайм.
- Функция ValidatePassword() компонента **Настройка безопасности: Пользователь** теперь возвращает корректные значения, если проверяемый пароль пустой или содержит пробелы.
- Исправлена ошибка, из-за которой права, назначенные роли, не удалялись при использовании функции RoleDeleteRight() компонента **Настройка безопасности: Приложение**.

Изменения документации

Редакция 1

Обновлен пример возвращаемого значения:

- для функции GetMembersList() компонента 2.9. Настройка безопасности: Группа (SecurityManagerGroup) (стр. 108).

- для функции **RequestUsersList()** компонента 2.6. Настройка безопасности: Менеджер (SecurityManager) (стр. 76). Пример приведен в описании события RequestUsersListComplete (стр. 82), активирующегося в случае успешного выполнения функции.

Редакция 2

В разделе 1.6. Контроль целостности (стр. 33) руководства пользователя:

- актуализировано описание настройки контроля целостности со стороны SePlatform.Security;
- добавлен новый подраздел, в котором приведена расшифровка сообщений о проверке целостности файлов (стр. 40).

В справочном руководстве актуализировано описание свойства IC_List_JSON (стр. 64) компонента **Настройка безопасности: Контроль целостности**.

Редакция 3

Содержимое документа не изменилось.

Редакция 4

В справочном руководстве описан новый компонент - 2.11. Информация лицензирования: Получение (LicenseInfo) (стр. 135).

Редакция 5

В руководстве пользователя:

- В раздел 1.2. Подготовка к работе (стр. 6) добавлены команды установки и удаления веб-версии расширения на ОС Linux.
- Обновлен проект-пример, прикладываемый к разделу 1.3. Управление доступом в проектах (стр. 9).
- Обновлены скриншоты.

В справочном руководстве:

- Описана ранее не описанная функция компонента **Настройка безопасности: Пользователь** - ValidatePassword() (стр. 100). Здесь же актуализировано описание функции **SetPassword()**.
- Обновлен пример значения, возвращаемого функцией GetTokensList() (стр. 89) компонента **Настройка безопасности: Приложение**.

Редакция 6

Внутренние изменения. Содержимое документа не изменилось.

1.1

1.1.3

Улучшение

Функция **RequestUserList()** компонента 2.6. Настройка безопасности: Менеджер (SecurityManager) (стр. 76) теперь предоставляет больше данных. Раньше список, возвращаемый функцией, содержал только uid и логин каждой учетной записи подсистемы безопасности. Теперь список содержит больше данных: фамилии, должности, номера телефонов и пр.
Исправление
Компонент 2.1. Контекст безопасности (SecurityContext) (стр. 47) больше не посылает в журнал приложений сообщения о внутренней работе. К таким относились сообщения вида "Деструктуризация булевского токена ...", "Удаление указателя на прокси контекст у токена ..." и т.п.

Изменения документации

Редакция 2

Описан ранее не описанный компонент **`Мастер конфигурирования Security`**:

- В руководство пользователя добавлен новый раздел 1.7. Конфигурирование агента безопасности (стр. 41), описывающий пример использования компонента в проектах SePlatform.HMI.
- В справочное руководство добавлено описание свойств и методов компонента.

Редакция 3

В руководстве пользователя:

- В раздел 1.2. Подготовка к работе (стр. 6) добавлена информация об установке и удалении SePlatform.HMI.Security на ОС Linux.
- Обновлен проект-пример в разделе 1.5. Собственный конфигуратор подсистемы безопасности (стр. 22). Для всех форм проекта создан общий **`Контекст безопасности`**. Он помещен в глобальные объекты.

В справочном руководстве:

- Обновлен пример значения, возвращаемого функцией **RequestUserList()** компонента 2.6. Настройка безопасности: Менеджер (SecurityManager) (стр. 76). Пример приведен в описании события RequestAppListComplete (стр. 81), активирующегося в результате успешного выполнения функции.
- Добавлена информация об особенности события **CurrentUserChanged** (стр. 58) - оно активируется дважды при запуске проекта SePlatform.HMI в рантайм.

Также исправлена история изменений для второй редакции документа, соответствующей SePlatform.HMI.Security версии 1.1. Ранее было указано, что содержимое документа не изменилось.